This is the author's copy of chapter 9 as published in [1]. The complete book can be found at https://www.elsevier.com/books/immersive-video-technologies/valenzise/978-0-323-91755-1.

CHAPTER

9 Light field processing for media applications

From practical challenges to neural rendering

Joachim Keinert[a], Laura Fink[a], Florian Goldmann[a], Muhammad Shahzeb Khan Gul[a], Tobias Jaschke[a], Nico Prappacher[a], Matthias Ziegler[a], Michel Bätz[a], and Siegfried Fößel[a]

[a] *Fraunhofer Institute for Integrated Circuits IIS, Am Wolfsmantel 33, 91058 Erlangen, Germany*
<firstname>.<surname>@iis.fraunhofer.de

ABSTRACT

Light fields describe the amount and direction of light rays flowing through every point in space. They allow reproducing different perspectives of a captured scene or a digitized object. This is particularly beneficial for photo-realistic virtual reality scenarios. Unfortunately, practical applications are still rare, as capturing and processing of information divided up in rich angular and spatial details contained in a light field is difficult. To overcome this situation, this chapter will describe algorithms and software solutions for both sparse and dense light fields. At first, a complete pipeline from capturing and processing to rendering of light fields is described. Light field capture is performed using a custom-built camera array. A node-based processing solution in a professional post-production environment permits to steer the light field reconstruction process for best achievable quality. The result can then be integrated into a VR environment with the help of a game engine. In the second part of this chapter, possibilities of neural networks to increase achievable rendering quality or reduce processing and capturing efforts are explored in detail.

KEYWORDS

light field processing and rendering, depth image-based rendering, geometric calibration, real-time rendering, neural light field algorithms, post production

9.1 Light field processing chain overview

Using the light field data captured by camera arrays for a practical application requires an algorithmic processing chain that allows to interpolate between the different views. This chapter focuses on multi-camera setups without active depth measurement. It is assumed that all cameras are synchronized precisely and capture a dynamic scene at interactive rates, i. e., larger than 30fps.

Fig. 9.1 depicts three possible processing pipelines. Image acquisition and camera calibration is similar for all of them as precise calibration data is mandatory for all

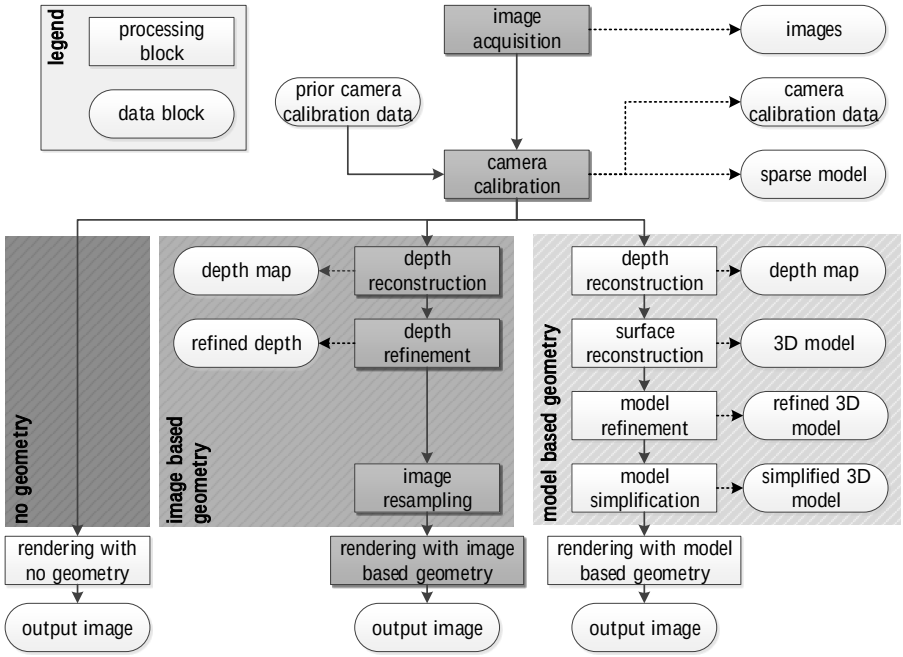


FIGURE 9.1

Multi-camera image processing pipelines for novel view synthesis from dense (left) and sparse multi-camera setups (center & right). Solid lines indicate consecutive processing steps; dotted lines represent results of a specific processing step.

subsequent steps of a reconstruction pipeline. Major considerations for the design and implementation of suitable calibration methods will be discussed in Section 9.2.

The leftmost path of Fig. 9.1 depicts the case when a densely sampled light field is available. In this case, re-sampling the plenoptic function results in the desired novel view irrespective of any geometric information. Therefore, this type of rendering is also known as “rendering without geometry” or "image-based rendering" [2–4]. As the spatial sampling rate needs to be much higher than the physical spacing between the individual cameras permits, this approach is only suitable for static scenes and not for dynamic scenes. Instead of a regular camera array, a camera mounted on industrial grade robotic arms can be used to capture a static scene with very high density [5, 6].

For sparse multi-camera setups, one of the central or rightmost paths of Fig. 9.1 labeled with "image based geometry" or "model based geometry" can be applied. The central path depicts a processing pipeline that renders novel views using image based geometry represented for instance by depth or disparity maps. Here, depth reconstruction and optional depth refinement form the major processing steps. Typically, the reconstruction step estimates an initial depth map for each input view using a set of adjacent views as input. Pairwise stereo matching [7] yields a set of individual

depth maps per view that need to be normalized and finally merged into a single depth map. Subsequently, the resulting depth maps can be refined using a variety of filtering steps. Such filtering may include occlusion checks that test geometric consistency of this image based 3D model across a set of input views. Inconsistent pixel values can be discarded and need to be concealed using methods such as cross-bilateral or cross-percentile filters [8, 9]. The resulting refined depth maps should provide a depth value for every pixel to avoid missing information in rendered images.

The rightmost path of Fig. 9.1 uses 3D models for geometry instead of images. In this branch, a geometric 3D model, e. g., using surface reconstruction techniques [10], is reconstructed. These approaches are able to close possible holes in the initial depth maps allowing those to be sparser than for the previous methods. Surface reconstruction aims to obtain a visual hull of the underlying scene.This is achieved by projecting input disparity maps into a 3D point cloud, a subsequent fusion of nearby points, and finally the creation of surface polygons [11]. Afterwards, this model (e. g., a mesh) forms the basis for subsequent refinement and simplification steps.

The image acquisition and camera calibration shown in Fig. 9.1 is common to all these paths and is explained in the following section.

9.2 Image acquisition and geometric camera calibration

The overall image quality of any output image described above does not only depend on the processing chain itself, but also on the quality of the input images. Besides precise synchronization of all incorporated cameras including trigger, exposure, gain and other parameters, colorimetric calibration might be required [12–15].

In addition to those pre-processing steps, efficient depth reconstruction using stereo matching techniques requires precisely aligned images ("rectified images") as it is assumed that corresponding pixels can be found in the same line of a horizontally adjacent image or in the same column of a vertically adjacent image. Even a slight misalignment can significantly degrade the quality of an estimated stereo disparity map. As the required accuracy cannot be achieved solely by mechanical adjustment, extrinsic and intrinsic camera parameters need to be determined [16]. Based on this parameter set, rectifying homographies can be created for a whole set of cameras (in case of a planar camera array) or individually for each pair of cameras. In the resulting rectified images, corresponding points are located in the same row for horizontally neighbored cameras, and in the same column for vertically adjacent cameras.

Determining the camera parameters requires a suitable camera model as well as the basics of projection and re-projection. An overview of the image formation process and camera models has been given by Schoeberl [17]. The concept of multiple view geometry [18] describes the relations between corresponding pixels showing the same object points in different images captured with one or several (pinhole) cameras. Based on these principles, many authors have proposed algorithms that determine position, orientation, focal length, lens distortions and other parameters of one or

several cameras [19–29]. The reconstruction problem is typically formulated as a non-linear optimization that minimizes the re-projection error between an observed point and its re-projected 3D point.

The highest precision as well as a link to absolute metric units can be obtained from active calibration methods that employ calibration objects such as checkerboards [25, 29]. However, even slight modifications to the camera setup can invalidate a parameter set, hindering a later post-processing. Thus, active calibration techniques are not optimal for simple use. In contrast to those active methods, Structure-from-Motion (SfM) pipelines [30], for instance, do not require prior knowledge about the objects contained in the scene. Hence they can compute both intrinsic and extrinsic parameters from almost arbitrary images as long as they contain recognizable features [31] and objects in different depths. To this end, SfM pipelines jointly reconstruct camera parameters and a (sparse) 3D model of a scene. This requires capturing a (static) scene from many different perspectives, preferably with a single camera.

For camera arrays, the estimation of the intrinsic and extrinsic parameters can be simplified in case of a precisely manufactured and robust mechanical framework. Thanks to a mechanically stable construction, first estimated values for the camera parameters can be derived from the structural design. They include the focal length of each camera, the ideal orientation as well as the ideal position of each camera. Due to different types of deviations such as improper focal length or orientation, the real configuration of any camera system will differ from these estimated value. Luckily, for some parameters such as the position of each camera, deviations to be expected are rather small. This knowledge can be translated into an additional constraint in a non-linear optimization problem reducing the possible degrees of freedom (prior camera calibration data in Fig. 9.1). In contrast to many other calibration techniques, such a method can directly optimize for the remaining vertical error in stereo pairs and neglects the position of a corresponding point in 3D space. Then, only the orientations, relative focal lengths, and principal points need to be determined or approximated [16].

Once the camera parameters have been successfully computed, the geometry of the scene can be reconstructed as described in the following section.

9.3 Depth reconstruction

Reconstructing a dense representation from a sparsely sampled light field by depth reconstruction techniques forms the central point of a light field processing chain for sparse camera arrays. For each pixel in an image, the goal is to find all corresponding pixels in all other views. The geometric relation between a pair of cameras allows to derive which pixels can be potentially related. Moreover, once the related pixels have been found, the distance between the cameras and the object point depicted by those pixels can be computed. This gain allows to interpolate missing camera views.

Unfortunately, depth reconstruction is an ill-posed problem: stereo matching assumes that a pixel (or a set of pixels surrounding the considered pixel) can be identified unambiguously in a secondary view, e. g., by computing the sum of absolute differ-

ences between a block in the primary view and all possible blocks in the secondary view [7]. Such a *cost function* determines the similarity of two pixel sets. In an ideal world, only the corresponding block in the secondary view will have zero difference. However, this assumption is only true if the scene is composed of Lambertian objects only and all surfaces are oriented directly towards the camera array. Otherwise, slight changes either in the viewing angle or due to perspective distortions may create false matches.

In addition to these constraints, occlusions need to be considered as well. A point, being visible in a primary view might be occluded by some foreground object in a secondary view. Thus, it is impossible to find the corresponding pixel. Hence, the cost function will select any other point that suits the optimization criterion (minimum cost function value) best.

Partially, pixels that do not have a valid correspondence in a secondary view due to occlusion can be detected by occlusion testing: considering a pixel in the primary view and its estimated correspondence in a secondary view, the estimated correspondence of the pixel in the secondary view should point back to the considered pixel in the primary view. Otherwise, the estimated correspondence is likely to be wrong.

In regard of these challenges, camera arrays benefit from the fact, that not only one pair of cameras is available for correspondence search. For a single input view, a plurality of possible secondary views are available to form a stereo pair. For example, a camera in the corner of an array has one horizontal, one vertical and one diagonal directly adjacent neighbor. A pixel in the primary view might be occluded in one of the secondary views, but might be visible in another secondary view so that a proper matching can still be performed. It is hence not important to interpolate pixels that have been discarded due to consistency problems as other stereo pairs can be selected to provide another set of reliable pixels that fill up unknown pixels. This procedure is also known as disparity merging [32, 33].

Stereo matching performs best if the underlying scene is well structured with natural textures that provide high contrast and uniqueness. "Flat" areas such as homogeneous backgrounds and overexposed areas need to be avoided when capturing a scene. A detailed introduction into this topic can be found in [7]. Looking at the tables from well-known benchmarks like Middlebury [34] or KITTI [35] reveals that the best-performing method often differs from scene to scene. In recent years, state-of-the-art algorithms have increased in complexity [36–38]. Slight parameter changes might improve the result on one scene, but could have a negative effect on another. For instance, such parameters comprise the block size of a block matcher, the threshold of a consistency test, or the kernel size or shape of a filter. In a practical application, it is thus important to quickly find a proper set of parameters that suits the current conditions of the scene and the cameras best (see also Section 9.4).

Once the initial disparity maps have been found, they can be further refined using for instance bilateral filtering [8, 9]. Interestingly, experiments have shown that the quality of a refined disparity map does primarily not depend on the complexity of the initial stereo matching method, but more on the following refinement steps [39]. In special cases, stereo matching based on a very simple block matching algorithm

can outperform more complex stereo matching methods such as ADCensus [40] or Semi-Global Matching [41]. This is also beneficial from a computational perspective: instead of executing a complex algorithm N times to obtain N initial results before refinement, a simple yet computationally effective algorithm may save energy and speed up the overall computation.

However, such an approach is only feasible in case a software environment allows to fine-tune the parameters of the algorithm for a specific scene. The following section provides some insights on such an interactive processing.

9.4 Interactive light field processing

As any modern media production requires some sort of post-processing, the employed camera array as well as the implemented software need to be compatible with standard production workflows. For example, this means that operations such as green-screen keying or color grading need to be applied to all input streams. In the best case, one set of parameters is sufficient to optimize the output of the keying operations. In the worst case, however, the parameters need to be adjusted individually for every input stream (e. g., if the raw colors differ from camera to camera).

This means that the underlying software needs to be capable of managing and processing multiple input streams. In a later step, more complex visual effects such as virtual backgrounds might be added or the processed material is being integrated into a larger 3D world. All this is only possible, if the output of the different algorithms can be understood and controlled by a user such as a post-production artist.

In this regard, traditional signal processing methods excel compared to AI-based methods explained later on in Section 9.7 as those typically encode the light field in a black box manner that is hardly accessible or adjustable with conventional tools. By allowing for an interactive processing, an experienced user (e. g., a post-production artist) can judge the obtained results based on his know-how. This user may then modify and optimize the processing steps such that the final result meets their requirements. Clearly, this requires a software system that supports changes to the pipeline and parameters. In addition, the software needs to provide intermediate results and interactive modification options to a user.

Node-based software systems such as Nuke [42, 43], Fusion [44, 45], or Blender [46, 47] can serve these needs in a favorable manner. By default, such software contains nodes for standard image processing tasks such as color correction, green-screen keying, image filtering, image transformation, and more. In addition, they support many different input and output formats and can often be extended by custom-designed plug-ins. The basic functionality of this software in combination with additional plug-ins forms an excellent basis for a flexible and user-configurable depth reconstruction pipeline.

Realception [16, 48], a plug-in suite for Nuke and Unreal Engine [49, 50] developed by Fraunhofer IIS, exemplifies such a node-based processing pipeline designed for scene-adapted depth reconstruction from multi-camera arrays. In addition to depth estimation and refinement operations, Realception also provides functionalities for

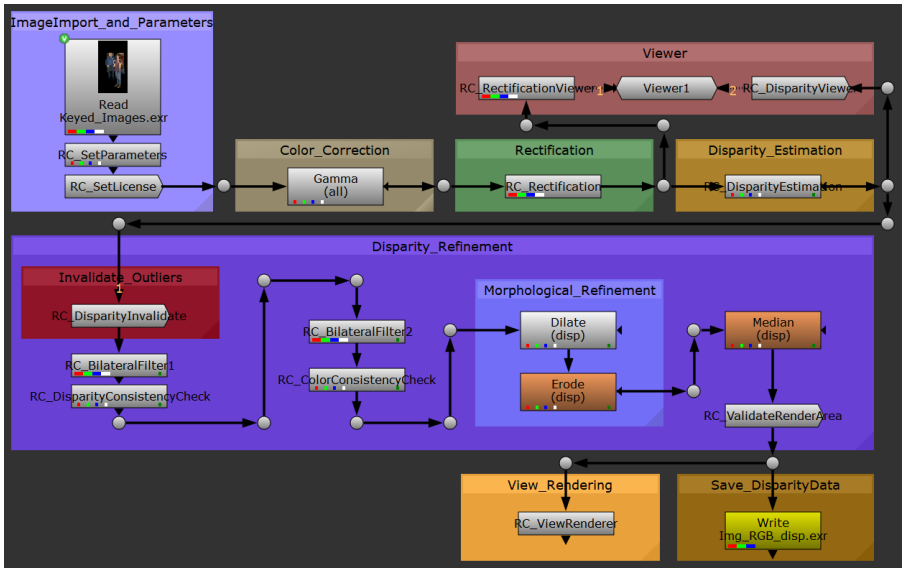


FIGURE 9.2

Example processing node graph for interactive light field editing within Nuke [42, 43] using the plug-in Realception [16, 48]. Initially, this graph reads a set of color images. Step by step, the pipeline reconstructs and refines a disparity map for each input image and renders a novel view.

image rectification and novel view synthesis. Nodes for cross-bilateral filtering or cross-percentile filtering [8, 51] as well as nodes for geometric and photometric consistency tests allow the improvement of the depth maps. Fig. 9.2 shows an exemplary pipeline in Nuke where all nodes (grey rectangles) with an RC-prefix denote Realception components. The pipeline first performs some pre-processing steps and reconstructs and refines disparity maps. Finally it renders a novel view from a light field input dataset. The principles to consider in this last step are discussed in the subsequent section.

9.5 Light field rendering

Light field rendering or view synthesis is the process of generating a target view from the captured input views. View synthesis can either be done by warping the pixels of the input views to the correct position for the target view, or by first transforming the input views into an intermediate representation (see Fig. 9.1). A 3D mesh as geometry model introduced in Fig. 9.1 combined with a corresponding texture atlas defining the color of each mesh point is an example for the latter. Novel views can then be computed by projecting the textured geometry to the target view according to the pinhole camera model. While this is heavily applied in computer graphics,

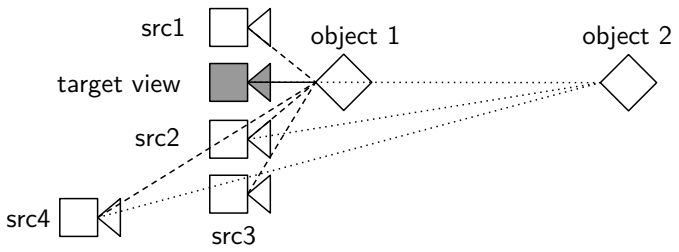


FIGURE 9.3

Selection of source cameras and pixels. Every ray corresponds to a pixel.

light fields can preserve the view-dependent appearance. In other words, an object point can change its color depending on the viewing direction which it is observed from. Consequently, the texture needs to be view-dependent as well, leading to more complex textures in the form of surface light fields [52, 53].

Alternatively, a texture less 3D geometry model can be used to identify for every target pixel the associated source pixel showing the same object point as illustrated in Fig. 9.3. These source pixels can be determined by projecting the intersection point of the target ray with the geometry model back into the source views. The selected source pixels need then to be blended into the target pixel value. Different object locations (object 1 and 2 in Fig. 9.3) lead to different selected source pixels.

In case only an image based geometry is available, source pixels can still be warped to the correct target view pixel based on depth or disparity maps. Together with the intrinsic and extrinsic camera parameters, they define the location of the depicted pixel point in 3D space. Finally, if no geometry information is available, such information needs to be hypothesized. This however will quickly lead to rendering artifacts in case the distance between the cameras is too large.

Thus, whatever rendering strategy is chosen, a light field view synthesis implementation needs to determine which source cameras should be selected to generate the appearance of a desired target pixel. This is discussed in more detail in the following section. Section 9.5.2 will then investigate which target view positions are admissible to avoid missing information in the synthesized image.

9.5.1 Camera selection and blending

As pixel warping requires computation, efficient rendering algorithms try to select relevant source images beforehand and neglect all remaining pixel information. The first criterion to apply for selecting source cameras to be used for view synthesis is frustum visibility. A camera whose field of view frustum does not intersect with the desired target object point can be excluded from the set of possible source views for this object point. Please note that this can only be decided when the location of the target object in 3D space is known.

Secondly, the source view should provide sufficient resolution for the desired target view. Assuming identical intrinsic camera parameters (focal length, pixel size, number of pixels, etc.), camera *src4* in Fig. 9.3 cannot provide enough details for *object 1*, simply because the distance between the object and the source camera is much larger than the distance between the object and the target view.

Thirdly, as the appearance of an object point can change depending on the target viewing direction, the angle between the source ray and the target ray should be as small as possible. In case of general source camera placement, this decision depends on the location of the object and the cameras in 3D space. For example, while camera *src4* in Fig. 9.3 has a smaller angle with the target ray than camera *src2* for *object 1*, it is the opposite for *object 2* despite corresponding to the same target ray. Only when the source views are located on a convex or concave surface and a target ray does not intersect this surface more than once, the source camera selection based on the viewing angle is independent of the object location and only depends on the source and target view positions. This represents a great benefit for regular camera arrays. Please also note that for *src4* and *object 1*, the resolution and the angle criteria come to contradicting conclusions in regards of preference compared to camera *src2*. This makes camera selection a challenge.

Finally, all those cameras in which a desired object point is occluded need also to be excluded from the set of possible source views. Again, this can only be decided when a precise 3D model of the scene is available.

Given that the camera selection depends on the scene geometry, several view synthesis approaches use a proxy mesh during the rendering process. This holds in particular for more irregular camera placements during capture [2, 3, 54]. For video applications, this would however cause a huge computational burden, as mesh creation is still computationally intensive. Consequently, in such applications more regular camera placements are employed, simplifying both frustum visibility, resolution, and angular camera selection [55–58]. Only the occlusion analysis still requires a 3D model. This can be avoided by rendering from all available source views, and then discarding not needed ones on an individual decision for each target pixel. Obviously, this comes with a significant computation cost when the number of source views is large.

9.5.2 Observation space for planar arrays

Even when following the above principles, the target views cannot be placed arbitrarily in 3D space. Instead, they need to be located in an admissible observation volume as exemplified for a single object point and a planar light field array in Fig. 9.4. Supposing that the object point intersects the field of view frustum of every camera, the possible target views need to be located in a pyramid whose cone end is defined by the object point. Target views located outside of this pyramid correspond to an extrapolation, which is prone to disocclusions and hence rendering artifacts in case of insufficient inpainting. Please note that according to the previous section, target cameras should not be closer to the object point than the source cameras to avoid

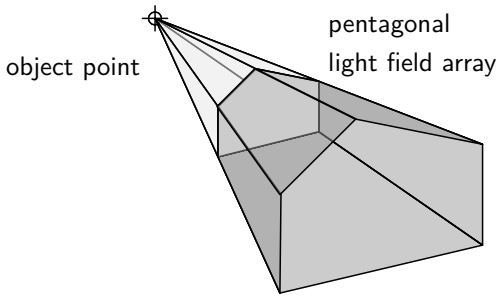


FIGURE 9.4

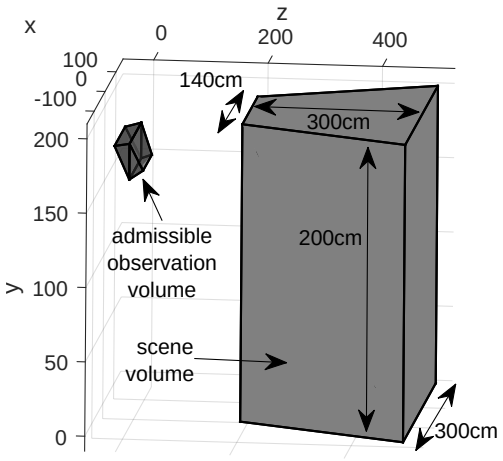
Admissible target view volume for a single object point and a pentagonal planar light field array. Each corner corresponds to a camera position.

a resolution loss. In other words, target cameras should be situated in the pyramid part that is shaded in a darker tone in Fig. 9.4 in case target views with insufficient resolution are to be avoided.

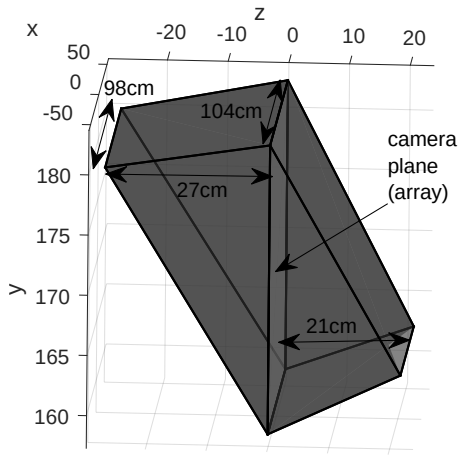
Having not only a single object point, but a complete volume containing the objects of the scene, the admissible observation volume for the target views can be computed by intersecting the pyramids whose cone ends are the extreme points of the scene volume. In practice, this can be done by means of software libraries such as [59] that are able to compute polytopes based on inequalities.

Fig. 9.5a depicts a corresponding example for a scene located in a volume with a height of 2 m, a depth of 3 m, a front width of 1.4 m and a back width of 3 m. The minimum distance between the camera array contained in the admissible observation volume and the scene is 2 m. The vertical position of the camera array has be set in such a way that an observer whose eye height equals 180 cm can move in negative z-direction as far as possible. In the shown example, this equals to -27 cm as depicted in Fig. 9.5b. It shows a magnification of the admissible observation volume. In case the target view position moves more into negative z-direction, this induces a risk of occlusions.

In summary, light field rendering requires both a careful capture and rendering setup. How this can be achieved in practice in a real-time manner is subject of the next section.



(a) Example scene volume with a depth of 300 cm and a volume height of 200 cm. The front width of the volume is 140 cm, the back width 300 cm.



(b) Admissible observation volume for a camera array with 3 rows, each having 10 cameras. The horizontal and vertical distance between the cameras equals 12 cm. The scene volume is located in positive z-direction in a distance of 2 m. The light-gray plane corresponds to the position of the camera array. Its height is 24 cm.

FIGURE 9.5

Admissible observation volume for a planar light field array.

9.6 Real-time rendering in game engines

With an increasing popularity of computer gaming, the wide-spread use of state-of-the art game engines such as Unreal Engine [49, 50] or Unity [60, 61] comes to no surprise. These engines are not only invaluable tools for the creation of new games, but rather have long surpassed their use as pure 3D game engines and continue to evolve beyond their original purpose. By now, game engines are a tool of choice for a plethora of digital use cases such as the creation of tech demos, architectural visualization, simulation, education, or as powerful 3D creation platforms for aspiring visual artists [62–67]. In the pursuit of ever-increasing graphical fidelity, methods for integrating photorealistic content into CGI environments are becoming more popular, photogrammetry being one of them [68]. This section will introduce two approaches for the seamless integration of light field footage into virtual environments by rendering new and physically coherent perspectives in real-time.

9.6.1 Pixel-based depth image-based rendering

As already discussed in the previous sections and outlined in Fig. 9.1, there are multiple ways for achieving a novel view synthesis when using multi-camera setups. In case a 3D geometry model is not available, pixel-based depth image-based rendering yields promising results [16, 39]. The possibility of high parallelization through GPU acceleration of the required pixel-based calculations makes this approach especially feasible for real-time applications such as virtual reality (VR) experiences. This allows for the integration and composition of real-world light field content into CGI environments where the necessary views are rendered in a perspectively correct fashion. The actual composition is realized by employing an intermediate screen element which displays a reconstructed view of the light field that is rendered w. r. t. the viewer's current position. The whole processing chain for this rendering approach consists of four components, as outlined in Fig. 9.6.

The input is expected to be N color images and corresponding disparity maps – one for each camera. In a first step, called forward warping, all disparity maps are warped to the camera pose of the target view (see Fig. 9.7, left image). This is done by calculating the pixel position of each source pixel on a new, empty map with the same dimensions as the source map and copying their respective disparity values, resulting in N warped disparity maps. Pixel positions that are outside of the image boundaries are discarded. Special consideration needs to be given to the fact that multiple pixels of a disparity map can be warped to the same target position. In practice, this is often the case when objects occlude each other. Since pixels with higher disparity values in the source disparity map correspond to objects that are closer to the camera, one straightforward approach is to prioritize the source pixel with the highest disparity value while discarding all others in case of a warping conflict.

After forward warping, an optional filtering step can be applied to improve the quality of the warped disparity maps by reducing artifacts such as speckles caused by missing or incorrect disparity values in the source maps. Also, since the target pixel

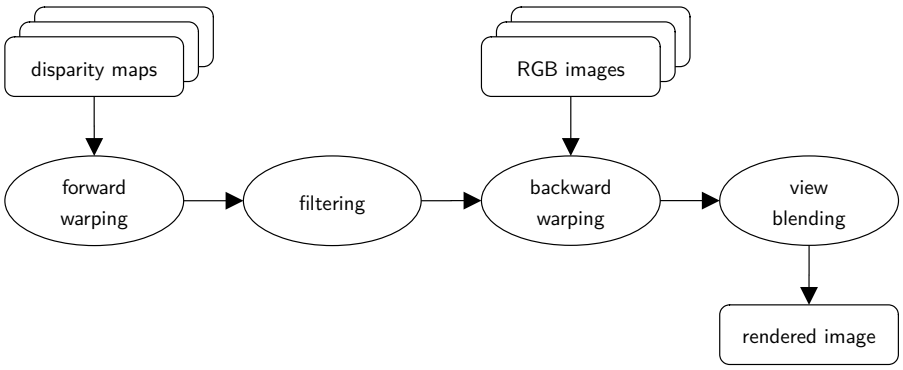


FIGURE 9.6

Overview of the pixel-based depth image-based rendering pipeline.

positions are discretized, gaps can occur when content is stretched by the warping process.

The following backward warping step can be seen as a texture fetching operation, as it relies on the forward warped disparity maps to get the color information for each target pixel by looking at the corresponding pixel locations in the source images. When attempting to access the color of a source image at a non-integer pixel position, an interpolation using the surrounding pixel values is conducted. The results are exemplified in Fig. 9.7 in the middle image.

After backward warping, one last blending step combines the backward warped color images to a single target view (Fig. 9.7, right image). Special consideration needs to be taken during the implementation of the blending step to ensure proper handling of occlusions and depth conflicts. Usually, not all N backward warped color images are used, but rather a selection of source views with the least Euclidian distance to the target position, as these views typically show less artifacts, interpolation errors, and occlusions compared to warped images from camera positions that are farther away from the render target. Since a distance-based cut-off criterion for source image selection leads to abrupt changes in the source view selection when the target position is moving through a scene (see Fig. 9.3), the application of a Gaussian distributed weight for each view is a more sensible approach. With this, smooth blending transitions between source view contributions are realized.

9.6.2 Implementation in Unreal Engine

The pixel-based depth image-based rendering approach as described in Section 9.6.1 can be implemented in Unreal Engine, while using GPU-accelerated compute shaders to facilitate real-time view rendering of video light field content for VR experiences [48, 69]. GPU acceleration of the rendering is crucial since the intended use in VR applications requires rendering with at least 90 fps, leaving a maximum time window

FIGURE 9.7

Left: forward warped and filtered disparity map. Middle: corresponding color image after backward warping using one source camera. Right: final render result after blending multiple backward warped color images together [16]. Both middle and right images are shown before compositing with a potential background which would smooth the sawtooth-like object borders.

of 11.1 ms until the whole light field rendering pipeline has to be completed.

To enable a fast rendering with minimal delay, static memory blocks for both input (RGB images and depth maps) and output are set up in the form of texture buffers that new data can be streamed into or read from in an asynchronous way. One advantage of this texture buffer approach is the capability to also play back videos by constantly streaming new data into the input buffer. The implementation uses a cyclic double buffer, one for RAM and one for VRAM so as to enable smooth streaming from a comparatively slow (SSD) hard disk drive into the GPU memory. Since the render target is also a texture buffer, this buffer can be assigned as material for any object inside the CG world, enabling this object to serve as display for the most recently rendered view. This object can also be considered as a screen or a light field display inside the virtual environment. With minor adjustments to the computations done in the forward and backward warping, this object can have various shapes, e. g., a sphere or cylinder, and could then also be used to accurately display footage from non-planar arrays, capture domes, or free-hand footage of static scenes. But in the basic use case, when processing data recorded by a planar light field array, the object has the shape of a plane, resulting in a flat light field display with similar properties like a window.

One remaining challenge of this VR-based approach is that the user and consequently the pose of the rendered target view can be freely chosen. This requires a proper handling of render poses that are not covered by the source footage because they are outside the admissible observation volume (c. f. Section 9.5.2). This can

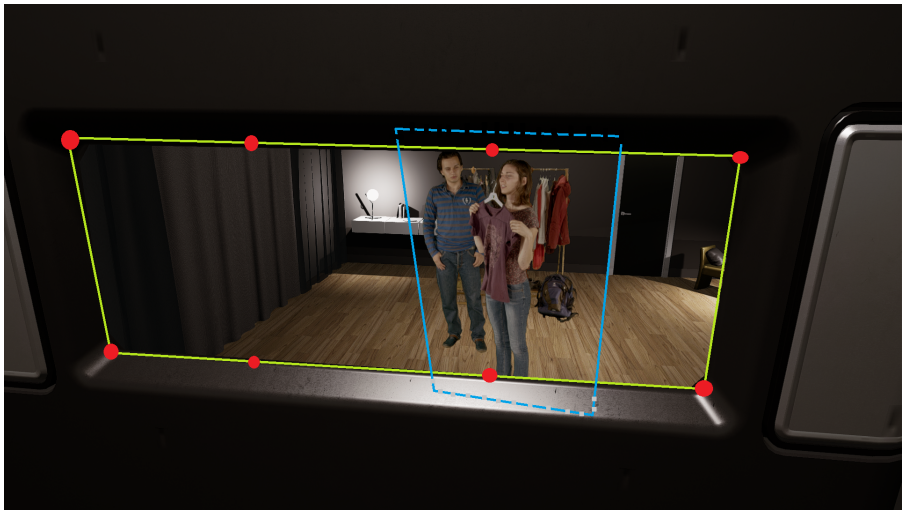


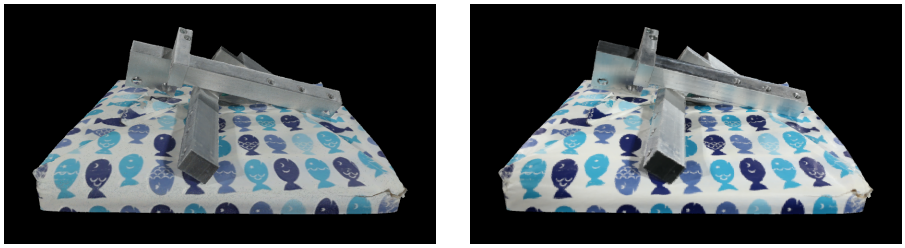
FIGURE 9.8

Real-time rendering of a light field dataset and integration into a virtual scene using Unreal Engine. The light field video data was captured by a planar 4×2 camera array; camera positions are marked in red. The wall around the window (green) blocks viewing angles from outside the admissible observer space. The blue rectangle surrounds the rendered content which is placed "inside" the virtual room [16].

be solved by placing a virtual wall or other opaque objects around the light field display. They naturally block viewing positions and angles from outside the admissible observer space, ensuring valid view renderings from any target pose. The light field display hence represents a window through which the captured content can be watched. This represents an intuitive way of blocking unwanted viewing directions and is easily understood by users, enhancing the feeling of immersion. Fig. 9.8 visualizes this concept with an example. The window analogy holds for all relevant aspects of the rendering, as long as the content displayed by the light field is physically placed behind the display, i. e., the light field display should be located between the viewer and the rendered content.

9.6.3 Mesh-based rendering for view-dependent effects

Render pipelines for meshes are highly optimized on GPUs to deliver remarkable performance for ambitious video game productions. Such pipelines can be extended with image-based rendering techniques for real-time applications to combine the benefits from both worlds for novel view synthesis. The open-source toolkit COLIBRI VR [70] aims to provide a mesh-based implementation for image-based rendering for VR content creators who use the Unity game engine. Any collection of photographs with either image or model based geometry representation (c. f. Section 9.1) from



(a) Diffuse texture **(b)** View-dependent texture

FIGURE 9.9

Comparison between diffuse texture map and view-dependent texture mapping using the unstructured lumigraph rendering from COLIBRI VR for the same mesh in the Unity game engine. Using the *diffuse texture*, the metal bars look flat and unreal. The *view-dependent texture* can better represent the Non-Lambertian surfaces.

COLMAP [30] can be transformed into virtual environments by the custom render pipeline. The main feature of this toolkit is to preserve view-dependent effects like specular highlights from the original images and with that, to achieve an accurate rendering of the reconstructed object surface beyond a diffuse texture map as shown in Fig. 9.9. Two different real-time rendering solutions are implemented for still photographs.

The first one is based on the unstructured lumigraph rendering algorithm [2] and requires a global mesh. The key element of this method is an angular and resolution-based blending function to achieve the view-dependent texture mapping. The mesh is being used as a geometry proxy to compute the blending weights. Source cameras with a location nearby a visible vertex and a viewing angle close to the target view will be assigned a higher weight compared with farther away or even occluded ones. The weights are computed in the vertex stage of the render pipeline (c. f. [71] for an overview of a conventional 3D graphics pipeline) for each source camera where all vertices of the mesh are being processed before the actual primitive assembly. In the fragment stage, the blending of the colors is done after the rasterization of the primitives to compute the final data for the rendered pixel.

To avoid interpolation of the weights between the different pipeline stages, both steps are additionally implemented as a standalone fragment shader which gives near ray tracing quality based on per-pixel weights. However, due to performance reasons, the actual number of relevant source cameras that contribute to the rendering are just a fraction of the original input and are mostly limited by hardware restrictions. A drawback of this rendering method are the visible ghosting artifacts caused by inaccurate camera calibration or mesh reconstruction.

The second implementation relies on per-view meshes using depth maps similar to [72]. A triangulation produces local meshes with mesh primitives covering several pixels. Those mesh primitives are generated in such a way that loss of accuracy

(a) Rendering from original local mesh **(b)** Rendering from filtered local mesh

FIGURE 9.10

Local mesh topology can be filtered to compensate stretching artifacts at disocclusion boundaries.

during view rendering can be avoided. As shown in Fig. 9.10, only at disocclusion boundaries, stretching artifacts are likely to appear which can be eliminated by the user with the toolkit.

For the rendering, the per-view meshes must be processed successively with a custom depth test to blend the projected colors at the target view. Otherwise, artifacts will appear at rendered occlusions if the depth order is not being considered. The blending weights are similar to ones used in the unstructured lumigraph rendering pipeline and therefore preserve to some extend view-dependent effects. The biggest challenge for this method is the geometric inconsistency that cannot be resolved on per-view basis and must be handled during the blending step.

9.7 Neural rendering and 3D reconstruction

After having introduced different light field processing and rendering pipelines using procedural signal processing, this chapter will focus on data driven methods. Section 9.7.1 starts by introducing learning-based architectures that follow the structure of procedural processing pipelines using an image based geometry representation. Afterwards, algorithms using implicit geometry representations are introduced in Section 9.7.2. These approaches differ more significantly from the concepts discussed so far, in that information is not stored in a discrete form anymore but as approximating functions. Finally, Section 9.7.3 shows how neural representations can also be applied to rendering concepts that rely on model based geometry.

9.7.1 Neural network-based rendering for light field angular super-resolution

This section will explain the image-based rendering (IBR) and depth image-based rendering (DIBR) approaches as illustrated in Fig. 9.1 which rely on neural networks. These approaches are particularly well suited to increase the limited angular resolution of plenoptic light field cameras [73].

Inspired by the success of deep learning for the task of single-image super-resolution (SR), different IBR methods for light field rendering emerged. Yoon *et al.* introduced two shallow convolutional neural networks (CNN), each consisting of three convolution layers, to sequentially interpolate both spatial and angular resolution [74]. Similarly, Gul *et al.* proposed utilizing the raw lenslet image correlation for spatial and angular resolution enhancement [75]. Exploring alternative representations, Wu *et al.* [76] and Wang *et al.* [77] reconstruct novel views of the light field by increasing the resolution of the epipolar images using a 2D CNN and a network consisting of both 2D and 3D convolutions.

DIBR methods synthesize novel views based on the recovered underlying geometry, such as depth. Kalantari *et al.* proposed the first end-to-end neural network-based method to mimic the conventional DIBR pipeline using convolutional neural networks [78]. The whole process is divided into two steps: depth estimation and color reconstruction. They proposed to reconstruct a dense light field using only four corner input images. First, they extracted features by calculating the mean and variance of the warped input images at discrete depth values. Then, a convolutional neural network processed these extracted features to estimate a disparity map at a novel target position. Based on the estimated geometry (i.e., disparity map), all four input views are warped to a target location which is then fed into another convolutional neural network synthesizing the final target view. As opposed to single view rendering in Kalantari's approach [78], Jin *et al.* [79] proposed a simultaneous reconstruction of all light field views. Following the two-step process, a CNN-based depth estimation module estimates the 4D light field depth maps to warp the input images to generate warped light fields. Then, a novel light field blending module composed of spatio-angular separable (SAS) convolutions [80] generates the final dense light field. A SAS convolution performs a 4D convolution by implementing a sequence of interleaved 2D convolutions (namely 2D spatial and 2D angular convolutions). In contrast to using four corner views, a dense 4D light field rendering from a single image is also possible; however, the quality of the rendered views suffers. Srinivasan *et al.* [81] estimate the 4D depth map from the center image using a nine-layer CNN consisting of dilated convolutions. Then a 3D convolution-based residual network renders the final light field by processing the warped light field generated using the estimated depth maps. Deviating from a two-step process, Navarro *et al.* [82] model the view rendering pipeline into three networks, namely feature extraction, depth estimation, and view selection. Rather than using the input views directly for depth estimation, they introduced a neural network to extract features from each image individually. In the end, a view selection network determines the contribution of each warped image

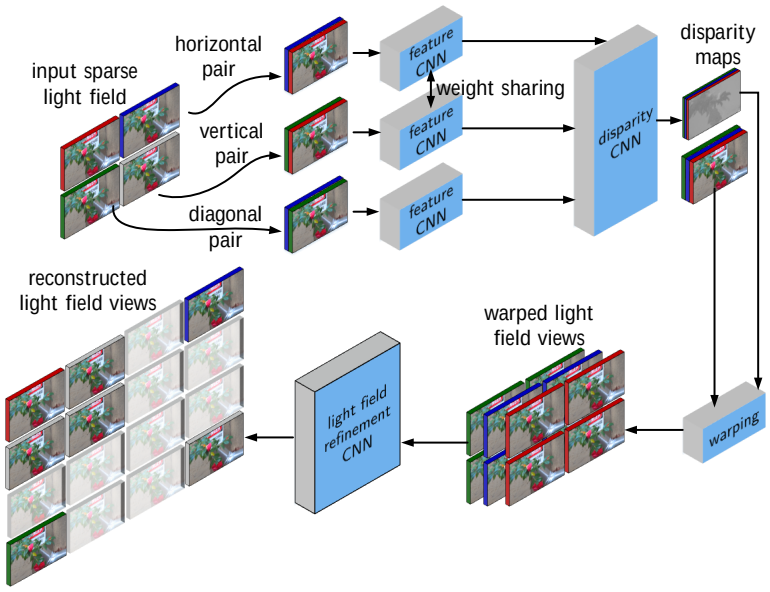


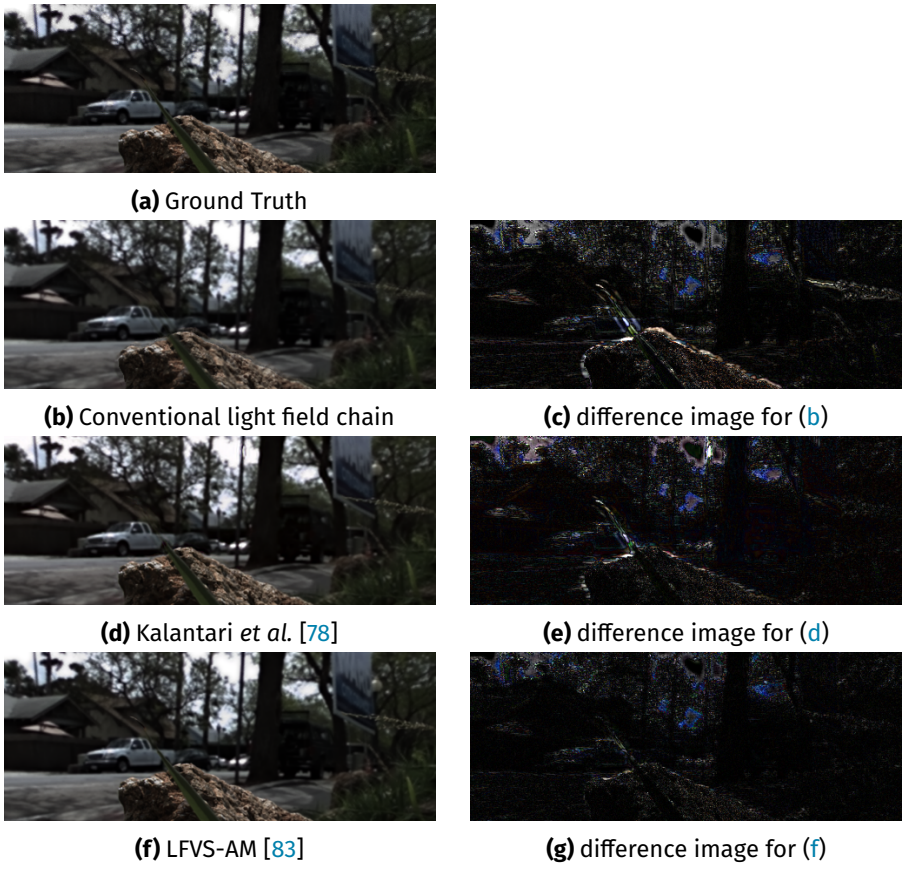
FIGURE 9.11

Pipeline of the light field view synthesis method *LFVS-AM* [83]. The process is divided into three CNNs: stereo feature extraction, disparity estimation, and light field refinement.

to the final result. However, these methods mostly fail to recover fine structures and occluded regions.

This can be improved by an attention mechanism as presented by *LFVS-AM* [83]. Similar to Navarro *et al.* [82], the *LFVS-AM* method also models the light field view rendering pipeline using three neural networks, stereo feature extraction, disparity estimation, and light field refinement, as shown in Fig. 9.11. The proposed method consists of three main features. First, *LFVS-AM* requires only three out of four corner views relative to the target view position, reducing the computational complexity. Second, *LFVS-AM* extracts features directly from stereo pairs which helps in estimating accurate disparity maps. Finally, an attention mechanism-based refinement network incorporating convolutional block attention modules (CBAMs) [84] reconstructs the novel views. CBAMs consist of spatial and channel attention focusing the network on critical regions of the features. Figure 9.12 shows the visual comparison of different DIBR approaches against the non-learning-based light field chain presented in Section 9.1.

Although these results show that neural networks can deliver good results, all the methods discussed above can only cope with light fields having small base lines. In case of larger base lines, the achievable quality significantly drops. One approach to solve this problem consists in the use of neural implicit representations as discussed



(a) Ground Truth

(b) Conventional light field chain

(c) difference image for (b)

(d) Kalantari *et al.* [78]

(e) difference image for (d)

(f) LFVS-AM [83]

(g) difference image for (f)

FIGURE 9.12

Visual comparison of different light field rendering methods using a test image from [78]. Except for *LFVS-AM* [83], all other methods failed to reconstruct the fine leave structure.

in the following section.

9.7.2 Neural implicit representations

In the previous section, neural networks were introduced into the regular depth image-based rendering pipeline. While this improved the final render quality, the neural networks are not able to overcome the shortcomings of the depth-based approach itself. Semi-transparencies, for example, are impossible to represent for depth-based approaches, since a single depth value per image pixel is not enough to represent multiple objects that may be visible to that pixel. To address the shortcoming of

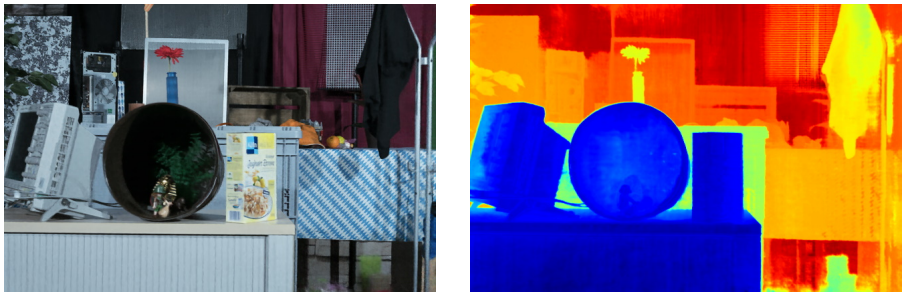


FIGURE 9.13

Results for novel view synthesis using NeRF [85]. Left: rendered image; right: false color depth map.

depth-based approaches, neural implicit representations have garnered more and more interest in recent years.

Specifically, Neural Radiance Fields (NeRF) [85], as introduced in [86], have shown promising visual fidelity on novel view synthesis tasks. The NeRF-generated depth map, as shown in Fig. 9.13, highlights a NeRF's capability to learn very detailed geometry for a given scene. Even fine details like leaves of plants that usually pose problems for traditional stereo matching are reconstructed properly when using a NeRF approach. Together with the volume rendering technique that allows NeRFs to learn semi-transparent objects and its capability to learn view-dependent effects, NeRF renders can achieve almost photorealistic quality, as can be seen in Fig. 9.13. The original NeRF publication sparked an explosion of follow-up works from the wider field of neural rendering. These works have further improved upon the original concept of NeRFs and reduced some of their downsides. These downsides include for instance their long training and rendering times, as well being limited to static scenes. While [86] has made a theoretical introduction to the concept of NeRFs, this section will introduce some of the aforementioned follow-up works, their trade-offs, and highlight some remaining practical challenges in the field of neural implicit scene reconstruction.

Practical challenge 1: input data

Like traditional light field rendering techniques, most neural implicit reconstruction approaches assume known camera poses for its input photographs. Since neural implicit approaches optimize a globally consistent geometry and appearance, even a slight misalignment in the poses would introduce conflicting information into the optimization process. While accurate pose estimates are no problem for computer-generated data, the concept of Structure-from-Motion (SfM) that is often used to register cameras for real scenes can fail or give subpar estimates. This would lead to reduced image quality of the final renders.

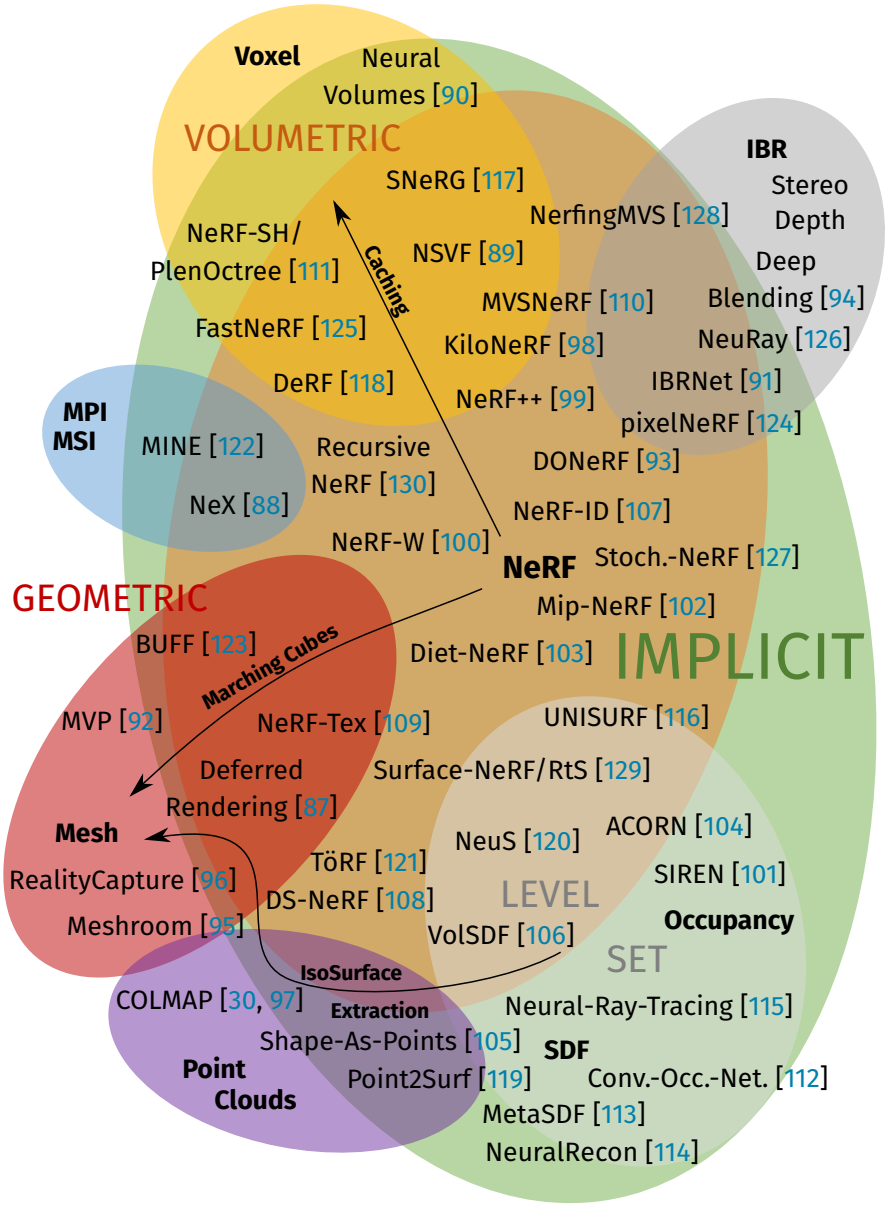


FIGURE 9.14

Overview map of the neural implicit representation literature [30, 87–130] surrounding neural radiance fields [85] as of November 2021. Dynamic, deformable, and re-lightable representations as well as representations with joint pose estimation have been omitted for clarity.

To avoid this, more recent works such as [131, 132] also allow the optimizer to backpropagate the rendering loss to the input camera poses. These are then jointly optimized during training. To allow faster convergence, the poses are usually still initialized with the output of a Structure-from-Motion estimation. The optimizer then only needs to fine-tune the pose estimates. In case no initial estimation is provided, these approaches can compensate for it with a more complex and longer training process. In either case, the final renders are of higher quality than they would be without fine-tuning the poses [131, 132].

Similarly, remaining distortion in the input photographs can also give the optimizer conflicting information, leading to degenerated solutions. Either camera distortions need to be modeled in the neural rendering pipeline or the input data for the training process needs to be undistorted beforehand. One neural representation that handles camera distortions in its training process is the Self-Calibrating-NeRF [133].

Since NeRF methods typically expect input photographs from a globally consistent scene, casually photographed scenes with slightly dynamic elements, such as leaves moving in the wind, can cause blurry reconstructions. Deformable neural representations have tackled this problem by introducing a neural deformation field into the rendering pipeline [134, 135]. This enables the remapping of points in space into a canonical reference space. The scene in this reference space is assumed to be static. The different input photographs and the slight perturbations in their underlying geometry can then be mapped into this canonical space via a neural network. Consequently, a globally consistent NeRF can be trained in the canonical space. The deformation field is trained jointly with the NeRF network.

Likewise, the deformation field can also be trained to map points from a starting position to a different timestamp in a dynamic sequence, e. g., an animation [136, 137]. The deformation field then learns how a scene changes over time. This allows dynamic neural representations to also represent light field video content. Note that the deformation field can readily represent deformation operations, but has trouble to model the sudden appearance or disappearance of objects, such as bursting bubbles. Also note that the literature on dynamic implicit representations is in its early stages, with most dynamic sequences being only few seconds long. The training times for dynamic representations are also significantly longer than for static scenes.

Moreover, the input data for an ordinary NeRF is expected to be consistently lit. Again, different shadows and lighting in the different input photographs would give the NeRF conflicting input information. The NeRF then usually tries to bake the different lighting conditions into the view-dependent appearance of objects, giving them excessive and unrealistic view-dependent effects. To reduce these problems, NeRF in the Wild (NeRF-W) [100] adds a learned latent embedding vector to all input photographs, that can encode lighting changes and many other inconsistencies. The NeRF-W method can then be trained on input photographs taken under vastly different conditions. Once trained, the embedding vectors can also be interpolated and allow NeRF-W to render a scene under different lighting conditions than those existing during capture. More recent works have also investigated how to model lighting more explicitly [138–140]. While most of them still expect studio-quality

input data, once trained, they can dynamically re-light the scene [138, 139] and even allow material editing [140].

Another less desirable way for an implicit model to deal with inconsistencies in the input data is through floater artifacts. Since the models are only trained to minimize the rendering error for the training poses, they could theoretically simply learn to place a small 2D photograph of the ground truth training view in front of the corresponding pose. The resulting reconstruction, however, would be degenerated and of little use as the model would be unable to correctly synthesize novel view points. Luckily, the regularization inherent in the neural networks discourages this kind of solution, as explained by [99]. Nevertheless, the above-mentioned floater artifacts can still appear. These are small, but dense artifacts floating in front of the training poses, that minimize the training error. When the NeRF training is presented with slightly conflicting information, e. g., inconsistent shadows, these floater artifacts allow the NeRF to still fit all input data. Recently, Mip-NeRF 360 has tackled this problem by using a specialized distortion loss [141].

Practical challenge 2: training

Another big drawback of most neural representations are their training times for days, even on high-end GPUs. The reason is that the NeRF network needs to be inferred for hundreds of samples per pixel during training. Note that a new model needs to be trained from scratch for every new scene.

As pixels can be rendered independently of each other, and samples along each ray can also be inferred independently of each other, an easy way to speed up rendering and thus training is parallelization. While the original NeRF approach makes limited use of such parallelization, a more recent JAX implementation [142] of NeRF, called JaxNeRF [143], scales the NeRF to multiple GPUs. This allows JaxNeRF to cut training time from days to hours, albeit at additional hardware cost.

Another way to speed up neural implicit training is the use of learned initialization. To this end, a meta-representation is trained on multiple scenes simultaneously, such that it can quickly be fine-tuned to any new scene [144]. This so called *meta-learning* is broadly used in the field of deep learning. The resulting meta-initialization itself does not represent any scene, but is instead used as a starting point for the training of new scene-specific representations. Since this starting point is optimized to converge towards novel scenes quickly, the training is faster than it would be from a random initialization. However, generating a meta-initialization itself is computationally intensive. Additionally, its ability to speed up training convergence depends on how close a novel scene is to the training distribution of the learned initialization.

Using explicit prior information, such as LiDAR point clouds, or point clouds from SfM pipelines have also shown promising results for speeding up training times [108]. Here, the depth values for the points of the point cloud are used as an additional supervisory signal that drives the so called DS-NeRF more quickly towards convergence. While this can speed up training by a factor of 2-6, DS-NeRF training still takes much longer for a given scene than traditional multi-view stereo approaches [11].

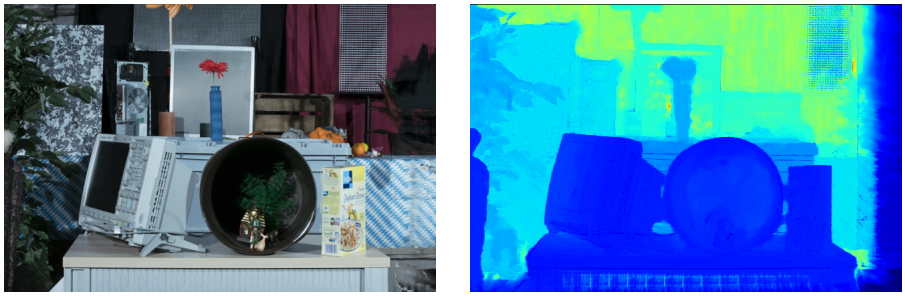


FIGURE 9.15

Results for novel view synthesis using a fine-tuned IBRNet [91]. Left: rendered image; right: false color depth map.

Other approaches, like IBRNet, require no scene-specific training at all. Instead, they learn prior information from a training set, also called *priors* in short. They can then be applied to novel unseen scenes [91, 124, 126]. Such approaches are hence said to *generalize* to unseen scenes. The learned prior information usually takes the form of trained neural network encoders and their feature extractors.They are trained on large 2D image datasets like ImageNet [145] or from large collections of existing scenes. As such, these approaches suffer from the same problems as a learned initialization: they require a computationally expensive initial training, before they can generalize to new scenes with no additional training. They also provide significantly worse geometric reconstruction, compared to, e. g., NeRF. This can be seen when comparing the depth map of an IBRNet in Fig. 9.15 with a NeRF depth map of the same scene in Fig. 9.13. It is also common to fine-tune these models for a given scene with a short additional training phase to gain a small quality improvement.

The high training cost of NeRFs is also limiting their potential resolution. While the NeRF algorithms make no corresponding assumption and could thus be used to train a NeRF on arbitrarily high input image resolutions, the training time scales at least linearly with the amount of input pixels. If the sample count along the volume rendering ray depth is increased with resolution, as is commonly done with multi-plane image approaches, the training time would scale cubically with the resolution. For high resolution content, NeRFs thus quickly become impractical. Most NeRFs in literature today are limited to less than FullHD resolutions.

Practical challenge 3: rendering

Another problem that limits the application of neural implicit representations is their slow rendering speed. Depending on the resolution, a single frame can take minutes to render.

Early attempts to speed up the rendering divided the 3D scene space into a regular grid of small voxels. Rendering can be sped up by one to three orders of magnitude,

when placing learned neural feature embeddings in a sparse subset of these voxels. Such a concept has been proposed by the Neural Sparse Voxel Fields (NSVF) [89]. Alternatively, a separate and smaller NeRF network can be learned for every voxel, as proposed by KiloNeRF [98]. However, both NSVF and KiloNeRF only work for bounded scenes. Unbounded and semi-bounded scenes, including forward-facing scenes, cause problems with the regular grid subdivision of the 3D space. In such cases, a space decomposition using frustrum shaped voxels (froxels) [146, 147] would serve better (see also [86]).

To facilitate real-time rendering, an implicit representation is usually converted into a format that is easier to render. Usually, this is a more explicit representation. After conversion, the original NeRF network can then be discarded. This new format can for example be a cached NeRF representation, such as a PlenOctree [111]. To cache a NeRF, it is densely sampled. Over-sampling may be applied for anti-aliasing purposes. The obtained outputs are stored in a grid representation such as an octree. Instead of performing a costly inference of the NeRF network multiple times per pixel during rendering, the PlenOctree can simply look up the NeRF output values in the cached octree. Similar to Neural Sparse Voxel Fields (NSVF) which exploit sparsity for a speed-up, the PlenOctree also only stores a sparse grid of output values. This avoids the cubic growth of required memory with increasing resolution. Still, most cached representations in literature are limited to less than FullHD resolutions.

Since NeRFs are five-dimensional to allow rendering view-dependent effects (three dimensions for geometry and a base color and two dimensions for view-dependent effects), naively caching the output of a NeRF would require a five-dimensional data structure. The curse of dimensionality would make it practically impossible to cache a NeRF with a very high accuracy. To avoid this problem, the cached values are independent of the actual viewing direction. The view-dependent effects are encoded in these outputs to still allow for instance the rendering of specularities. In the case of PlenOctree, the outputs of a NeRF-SH [111] are cached in the form of spherical harmonics (SH), which gives the methods its name NeRF-SH. Other caching approaches, like Sparse Neural Radiance Grids (SNeRG) [117], predict view-independent neural textures. These allow view-dependent effect reconstruction via deferred rendering.

The faster rendering of PlenOctree compared to the NeRF-SH, which it was extracted from, can also be used to accelerate training. To this end, a NeRF-SH training is stopped early and a PlenOctree is extracted. The PlenOctree is then fine-tuned with a similar rendering error towards convergence. Overall, this can significantly outperform a normal NeRF training [111].

Practical challenge 4: mesh extraction

Alternatively to caching, rendering can be sped up by converting a trained implicit representation into a mesh. Such meshes have been used in computer graphics for decades. Dedicated hardware and optimized rendering pipelines have made meshes a common choice for real-time rendering applications, such as video games. Extracting

a lightweight mesh from the detailed density field of a neural representation would allow real-time rendering, even on lower-end consumer devices.

The most simple way to extract a mesh from the density field of an implicit representation is a threshold-based marching cubes approach [148]. However, if the threshold is set too low or high, false-positive or false-negative geometry from artifacts might transfer into the extracted mesh. This is a known problem in literature [116, 120]. More sophisticated mesh extraction approaches have been tried in [123]. Still, many fine details from the neural density field are lost during mesh extraction.

To extract detailed meshes from optimized, neural representations, literature has started to incorporate level-sets into the volume rendering pipeline. Level-set modeling implicitly represents surfaces based on a function that maps a point in 3D space to a real value. All points leading to the same and well defined function value are considered to belong to the surface. A signed distance function (SDF), for instance, has a gradient norm of one with its zero level-set defining the surface [149].

Specifically, approaches like [106, 116, 120] convert a learned level-set function into a density field before volume rendering. This allows them to make use of the superior optimization process of density field approaches and drops the pixel mask requirement common in previous neural implicit level-sets [150]. Once the level-set function has been trained, isosurface extraction can generate a mesh with arbitrarily low quality loss from the level-set geometry. The only downside of level-sets compared to density fields is that they cannot readily support transparencies.

Once extracted, a neural texture for deferred rendering can provide view-dependent color appearance for the mesh geometry [87]. Alternatively, a texture can be extracted from the implicit representation itself. NeuTex [151] instead reformulates the training of the implicit representation such that it jointly learns a texture alongside the geometry. Furthermore, the rendering of the extracted mesh could be sped up by using one of the growing number of differential mesh renderers for mesh simplification, such as [152].

9.7.3 Neural textures

Using neural textures [87] for novel view synthesis enables high fidelity rendering of objects with complex appearance, including the reproduction of view-dependent effects. Moreover, they can conceal imperfections of the geometry itself without ever touching it, thus being complementary to the methods described in Section 9.7.2.

As the name of the method *deferred neural rendering* suggests, the approach adds a neural component to the traditional computer graphics technique *deferred rendering* [153, 154]. Deferred rendering is commonly used to reduce computations by splitting the rasterization of geometry and the actual shading into two passes. Thus, appearance computations like illumination and material evaluation are only performed for visible fragments.

In the case of deferred neural rendering, shading is performed by a neural network, often referred to as *neural renderer*. This neural renderer interprets neural descriptors that are projected to screen space via the sampling of the neural texture using standard

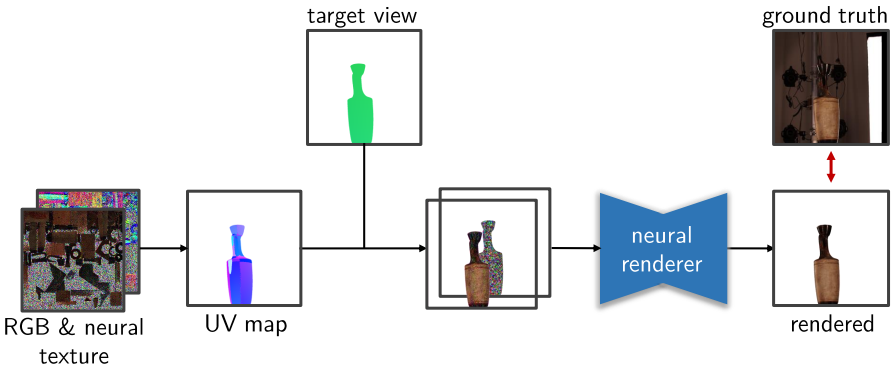


FIGURE 9.16

Deferred neural rendering pipeline using neural textures.

texture coordinates. By doing so, shading is performed from a statistical point of view rather than a physically-based one. An overview of the described pipeline is illustrated in Figure 9.16. Using traditional computer graphics techniques, the neural and RGB textures are projected into the target view. UV maps establish the relation between the texture images and the geometry primitives. The neural renderer then transforms the projected textures into a rendered image. Both the neural renderer and the neural textures can be trained by comparing the rendered image with the corresponding ground truth.

The neural renderer is often implemented with a U-Net-like [155] architecture [87, 156–158], as its skip connections are a favorable design choice for tasks that are framed by a 1-to-1 pixel correlation of in- and output. Thus, the architecture enables fast convergence and the reproduction of high frequency details.

Neural textures are tightly coupled to surface light fields [52, 53] and can be seen as a neurally encoded and tightly compressed form of such a light field. Surface light fields are parameterized by the surface position and the view direction – the position it is looked at from. For example, a surface light field may be implemented by storing multiple color samples for each valid texture coordinate, each sample corresponding to a different view direction, respectively. Novel view synthesis can then be performed by rendering the closest sample (with regard to position and angle) that can be found in the texture for a given view ray.

While there is not a neural descriptor stored per view direction, view dependency is achieved by the neural renderer which learns descriptor constellations. The render quality can be further enhanced by "masking out" unnecessary parts of the current view before the textured image is fed into the neural renderer network. To this end, Thies et al. propose the descriptors to be multiplied by the spherical harmonics coefficients of the first three bands of the per-pixel view direction. Thus, neural textures offer a vast compression factor compared to traditional surface light fields

that need to store an RGB value for a reasonable amount of directions for any given texture element position.

Note that this view dependency is not limited to view-dependent effects like reflections to mimic complex materials, but also fixes mismatching geometry – solely in screen space – up to some degree. This is an exceptionally interesting feature for novel view synthesis of real scenes as reconstructed meshes from photogrammetry are hardly ever perfect. Also, for highly textured materials like fur or grass which usually have to be modeled by volumes that are expensive to render, this ability is promising.

Using neural feature descriptors to encode information that go beyond standard RGB color is a paradigm seen quite frequently in the state of the art and proves their usability beyond mesh-based geometry representations. Similar to the mesh-based rendering, papers like [157–159] show how such descriptors can be used for point clouds in conjunction with a neural renderer to overcome sparsity and thus, provide great visual fidelity. Others [91] show the feasibility of descriptors for volume-based representations and alike.

Many works [87, 157, 159] on novel view synthesis propose overfitting to a single or in the best case to only a handful of somewhat similar scenes. Thus, retraining is needed for every new object or in the worst case even for the slightest change in the scene, limiting the use case to learning a single light field by rote. Scene re-composition, especially for objects with specular materials, proves hard as reflections are baked into the descriptors.

However, neural textures can be used beyond their use as compressed light fields. The generalization to unseen scenes, re-lighting, and scene editing is subject of current state-of-the-art methods [91, 156, 158, 160]. Meshry *et al.* [158] resolve occlusions and achieve re-lighting for touristic scenes by conditioning scene appearance on learned embeddings. Controllable re-lighting for indoor scenes is the subject in the works of Meka *et al.* [161] and Öchsle *et al.* [162]. Wang *et al.* [91] show promising results towards generalization and the applicability for various datasets. Another direction is the incorporation of the plenoptic function's time domain that is usually omitted and thus allowing for controllable animation of faces [163] or replaying video clips from novel view points [137].

In summary, neural textures and related work can be interpreted as highly condensed surface light fields allowing for high fidelity novel view synthesis. However, the direction of ongoing research points to tasks that go beyond the sole reconstruction of known scenes.

9.8 Conclusion and open challenges

Light fields offer a much richer representation of real world scenes in media applications as they enable observation from different perspectives. As a result, the experience is more immersive or realistic compared to a 2D image or video. To make this possible, it is necessary to design and implement a complete processing pipeline, encompassing light field capture, camera calibration, geometry reconstruction, con-

tent editing, and real-time rendering approaches. Many solutions have been proposed in this regard, making light field media experiences more and more realistic. Still, a lot of challenges need to be overcome to make light fields applicable in a larger scale. Core aspects include the achievable quality as well as the handling of the large amounts of data and necessary computation. Neural algorithms promise to bring a new way of thinking into these domains. While showing a very high potential to advance light field processing in a significant manner, they currently suffer from huge computation demands and low supported resolutions. Furthermore, a consolidation of the huge variety of possible solutions still needs to be achieved in the future.

Acknowledgments

Parts of this work have been funded by the Free State of Bavaria in the DSAI project and by European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 765911 (RealVision).

Bibliography

[1] G. Valenzise, M. Alain, E. Zerman, C. Ozcinar (Eds.), Immersive Video Technologies, Academic Press, 2022.

[2] C. Buehler, M. Bosse, L. McMillan, S. Gortler, M. Cohen, Unstructured lumigraph rendering, in: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), ACM Press, Los Angeles, CA, USA, 2001, pp. 425–432. doi:10.1145/383259.383309.

[3] A. Davis, M. Levoy, F. Durand, Unstructured Light Fields, Computer Graphics Forum 31 (2pt1) (2012) 305–314. doi:10.1111/j.1467-8659.2012.03009.x.

[4] Image-Based Rendering (IBR), in: K. Ikeuchi (Ed.), Computer Vision: A Reference Guide, Springer US, Boston, MA, 2014, pp. 399–399. doi:10.1007/978-0-387-31439-6_100163.

[5] F. S. Zakeri, A. Durmush, M. Ziegler, M. Bätz, J. Keinert, Non-Planar Inside-Out Dense Light-Field Dataset and Reconstruction Pipeline, in: IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 1059–1063. doi:10.1109/icip.2019.8803402.

[6] M. Ziegler, R. op het Veld, J. Keinert, F. Zilly, Acquisition system for dense lightfield of large scenes, in: 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), 2017 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), IEEE, Copenhagen, Denmark, 2017, pp. 1–4. doi:10.1109/3DTV.2017.8280412.

[7] D. Scharstein, R. Szeliski, R. Zabih, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, in: IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV), Kauai, HI, USA, 2001, pp. 131–140. doi:10.1109/smbv.2001.988771.

[8] C. Tomasi, R. Manduchi, Bilateral filtering for gray and color images, in: IEEE 6th International Conference on Computer Vision (ICCV), Bombay, India, 1998, pp. 839–846. doi:10.1109/iccv.1998.710815.

[9] S. Paris, P. Kornprobst, J. Tumblin, F. Durand, A Gentle Introduction to Bilateral Filtering and its Applications, in: ACM SIGGRAPH, 2007, p. 130. doi:10.1145/1281500.1281602.

[10] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, C. T. Silva, A Survey of Surface Reconstruction from Point Clouds, Computer Graphics Forum 36 (1) (2017) 301–329. doi:10.1111/cgf.12802.

[11] Y. Furukawa, C. Hernández, Multi-View Stereo: A Tutorial, no. 9,1/2 in Foundation and Trends in Computer Graphics and Vision, Now, Boston Delft, 2015.

[12] A. Ernst, A. Papst, T. Ruf, J.-U. Garbas, Check my chart: A robust color chart tracker for colorimetric camera calibration, in: 6th International Conference on Computer Vision / Computer Graphics Collaboration Techniques and Applications, MIRAGE '13, New York, NY, USA, 2013, pp. 1–8. doi:10.1145/2466715.2466717.

[13] N. Joshi, B. Wilburn, V. Vaish, M. Levoy, M. Horowitz, Automatic Color Calibration for Large Camera Arrays, Tech. Rep. CS2005-0821, UC San Diego: Department of Computer Science & Engineering, San Diego (May 2005).

[14] A. Ilie, G. Welch, Ensuring color consistency across multiple cameras, in: IEEE International Conference on Computer Vision (ICCV), Beijing, China, 2005, pp. 1268–1275 Vol. 2. doi:10.1109/iccv.2005.88.

[15] K. Li, Q. Dai, W. Xu, High quality color calibration for multi-camera systems with an omnidirectional color checker, in: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Dallas, TX, USA, 2010, pp. 1026–1029. doi:10.1109/icassp.2010.5495322.

[16] M. Ziegler, Advanced image processing for immersive media applications using sparse light-fields, Ph.D. thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen (2019).

[17] M. Schöberl, Modeling of Image Acquisition for Improving Digital Camera Systems, Ph.D. thesis,

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen (2013).

[18] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, 2nd Edition, Cambridge University Press, Cambridge, 2004. doi:10.1017/CBO9780511811685.

[19] F. Bajramovic, J. Denzler, Self-calibration with Partially Known Rotations, in: F. A. Hamprecht, C. Schnörr, B. Jähne (Eds.), Pattern Recognition, Vol. 4713 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 1–10. doi:10.1007/978-3-540-74936-3_1.

[20] P. Baker, Y. Aloimonos, Complete calibration of a multi-camera network, in: Proceedings IEEE Workshop on Omnidirectional Vision (Cat. No.PR00704), 2000, pp. 134–141. doi:10.1109/omnvis.2000.853820.

[21] M. Brückner, F. Bajramovic, J. Denzler, Intrinsic and extrinsic active self-calibration of multi-camera systems, Machine Vision and Applications 25 (2) (2014) 389–403. doi:10.1007/s00138-013-0541-x.

[22] A. Bushnevskiy, L. Sorgi, B. Rosenhahn, Multicamera Calibration from Visible and Mirrored Epipoles, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 3373–3381. doi:10.1109/cvpr.2016.367.

[23] X. Chen, J. Davis, P. Slusallek, Wide area camera calibration using virtual calibration objects, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, Hilton Head Island, SC, USA, 2000, pp. 520–527. doi:10.1109/cvpr.2000.854901.

[24] F. Kahlesz, C. Lilge, R. Klein, Easy-to-use calibration of multiple-camera setups, International Conference on Computer Vision Systems (ICVS) Camera Calibration Methods for Computer Vision Systems (CCMVS) (Dec. 2007). doi:10.2390/biecoll-icvs2007-177.

[25] G. Kurillo, H. Baker, Z. Li, R. Bajcsy, Geometric and Color Calibration of Multiview Panoramic Cameras for Life-Size 3D Immersive Video, in: International Conference on 3D Vision (3DV), Seattle, WA, USA, 2013, pp. 374–381. doi:10.1109/3dv.2013.56.

[26] T. Svoboda, D. Martinec, T. Pajdla, A convenient multicamera self-calibration for virtual environments, Presence: Teleoperators and Virtual Environments 14 (4) (2005) 407–422. doi:10.1162/105474605774785325.

[27] S. Urban, S. Wursthorn, J. Leitloff, S. Hinz, MultiCol bundle adjustment: A generic method for pose estimation, simultaneous self-calibration and reconstruction for arbitrary multi-camera systems, International Journal of Computer Vision 121 (2) (2017) 234–252. doi:10.1007/s11263-016-0935-0.

[28] V. Vaish, B. Wilburn, N. Joshi, M. Levoy, Using plane + parallax for calibrating dense camera arrays, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 1, Washington, DC, USA, 2004, pp. 2–9. doi:10.1109/cvpr.2004.1315006.

[29] Z. Zhang, A flexible new technique for camera calibration, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (11) (2000) 1330–1334. doi:10.1109/34.888718.

[30] J. L. Schönberger, J.-M. Frahm, Structure-from-Motion Revisited, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 4104–4113. doi:10.1109/cvpr.2016.445.

[31] J. Ma, X. Jiang, A. Fan, J. Jiang, J. Yan, Image Matching from Handcrafted to Deep Features: A Survey, International Journal of Computer Vision 129 (1) (2021) 23–79. doi:10.1007/s11263-020-01359-2.

[32] F. Zilly, Method for the automated analysis, control and correction of stereoscopic distortions and parameters for 3D-TV applications : New image processing algorithms to improve the efficiency of stereo- and multi-camera 3D-TV productions, Ph.D. thesis, Technische Universität Berlin, Berlin (2015). doi:10.14279/depositonce-4618.

[33] F. Zilly, C. Riechert, M. Müller, P. Eisert, T. Sikora, P. Kauff, Real-time generation of multi-view

video plus depth content using mixed narrow and wide baseline, Journal of Visual Communication and Image Representation 25 (4) (2014) 632–648. doi:10.1016/j.jvcir.2013.07.002.

[34] Daniel Scharstein, Richard Szeliski, Heiko Hirschmüller, Middlebury Stereo Vision Page, https://vision.middlebury.edu/stereo/ (Last accessed on 06.12.2021).

[35] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, The KITTI Vision Benchmark Suite, http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo (Last accessed on 06.12.2021).

[36] R. A. Hamzah, H. Ibrahim, Literature Survey on Stereo Vision Disparity Map Algorithms, Journal of Sensors (2016) 1–23doi:10.1155/2016/8742920.

[37] Z. Tu, W. Xie, D. Zhang, R. Poppe, R. C. Veltkamp, B. Li, J. Yuan, A survey of variational and CNN-based optical flow techniques, Signal Processing: Image Communication 72 (2019) 9–24. doi:10.1016/j.image.2018.12.002.

[38] M. Zhai, X. Xiang, N. Lv, X. Kong, Optical flow and scene flow estimation: A survey, Pattern Recognition 114 (2021) 107861. doi:10.1016/j.patcog.2021.107861.

[39] M. Ziegler, A. Engelhardt, S. Müller, J. Keinert, F. Zilly, S. Foessel, K. Schmid, Multi-camera system for depth based visual effects and compositing, in: 12th European Conference on Visual Media Production, CVMP '15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 1–10. doi:10.1145/2824840.2824845.

[40] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, Xiaopeng Zhang, On building an accurate stereo matching system on graphics hardware, in: IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 2011, pp. 467–474. doi:10.1109/iccvw.2011.6130280.

[41] H. Hirschmuller, Accurate and efficient stereo processing by semi-global matching and mutual information, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, San Diego, CA, USA, 2005, pp. 807–814 vol. 2. doi:10.1109/cvpr.2005.56.

[42] Nuke | VFX and Film Editing Software, https://www.foundry.com/products/nuke-family/nuke (Last accessed on 06.12.2021).

[43] Nuke (software), Wikipedia (Last accessed on 06.12.2021).

[44] Fusion 17 | Blackmagic Design, https://www.blackmagicdesign.com/products/fusion/ (Last accessed on 06.12.2021).

[45] Blackmagic Fusion, Wikipedia (Last accessed on 06.12.2021).

[46] Blender (software), Wikipedia (Last accessed on 06.12.2021).

[47] B. Foundation, Blender.org - Home of the Blender project - Free and Open 3D Creation Software, https://www.blender.org/ (Last accessed on 06.12.2021).

[48] Realception®, https://www.iis.fraunhofer.de/en/ff/amm/content-production/realception.html (Last accessed on 06.12.2021).

[49] Unreal Engine | The most powerful real-time 3D creation tool, https://www.unrealengine.com/en-US/ (Last accessed on 06.12.2021).

[50] Unreal Engine (software), Wikipedia (Last accessed on 06.12.2021).

[51] C. Riechert, F. L. Zilly, M. Mã, Real-Time Disparity Estimation Using Line-Wise Hybrid Recursive Matching and Cross-Bilateral Median Up-Sampling, in: International Conference on Pattern Recognition (ICPR), Tsukuba Science City, Japan, 2012, p. 4.

[52] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, R. Grzeszczuk, Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields, ACM Transactions on Graphics 21 (3) (2002) 447–456. doi:10.1145/566654.566601.

[53] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, W. Stuetzle, Surface light fields for 3D photography, in: 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), New Orleans, LA, USA, 2000, pp. 287–296. doi:10.1145/344779.

344925.

[54] P. Hedman, T. Ritschel, G. Drettakis, G. Brostow, Scalable inside-out image-based rendering, ACM Transactions on Graphics 35 (6) (2016) 1–11. doi:10.1145/2980179.2982420.

[55] A. P. Pozo, M. Toksvig, T. F. Schrager, J. Hsu, U. Mathur, A. Sorkine-Hornung, R. Szeliski, B. Cabral, An integrated 6DoF video camera and system design, ACM Transactions on Graphics 38 (6) (2019) 1–16. doi:10.1145/3355089.3356555.

[56] D. Bonatto, S. Fachada, S. Rogge, A. Munteanu, G. Lafruit, Real-Time Depth Video-Based Rendering for 6-DoF HMD Navigation and Light Field Displays, IEEE Access 9 (2021) 146868–146887. doi:10.1109/access.2021.3123529.

[57] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. DuVall, J. Dourgarian, J. Busch, M. Whalen, P. Debevec, Immersive Light Field Video with a Layered Mesh Representation, ACM Transactions on Graphics 39 (4) (2020) 15. doi:10.1145/3386569.3392485.

[58] B. Vandame, N. Sabater, G. Boisson, D. Doyen, V. Allié, F. Babon, R. Gendrot, T. Langlois, A. Schubert, Pipeline for Real-Time Video View Synthesis, in: IEEE International Conference on Multimedia Expo Workshops (ICMEW), London, UK, 2020, pp. 1–6. doi:10.1109/ICMEW46912.2020.9105988.

[59] M. Herceg, M. Kvasnica, C. N. Jones, M. Morari, Multi-Parametric Toolbox 3.0, in: European Control Conference (ECC), Zurich, Switzerland, 2013, pp. 502–510. doi:10.23919/ecc.2013.6669862.

[60] U. Technologies, Unity Real-Time Development Platform | 3D, 2D VR & AR Engine, https://unity.com/ (Last accessed on 06.12.2021).

[61] Unity (software), Wikipedia (Last accessed on 06.12.2021).

[62] F. Lv, C. Hu, Research on Simulation of pedestrian flow Unity 3D through Multiple Exit Architecture, in: International Conference on Computer Engineering and Intelligent Control (ICCEIC), Chongqing, China, 2020, pp. 51–54. doi:10.1109/icceic51584.2020.00018.

[63] F. Valls, E. Redondo, D. Fonseca, P. Garcia-Almirall, J. Subirós, Videogame Technology in Architecture Education, in: M. Kurosu (Ed.), International Conference on Human-Computer Interaction – Human-Computer Interaction. Novel User Experiences, Lecture Notes in Computer Science, Springer, Cham, 2016, pp. 436–447. doi:10.1007/978-3-319-39513-5_41.

[64] D. Chaves, J. R. Ruiz-Sarmiento, N. Petkov, J. Gonzalez-Jimenez, Integration of CNN into a Robotic Architecture to Build Semantic Maps of Indoor Environments, in: I. Rojas, G. Joya, A. Catala (Eds.), International Work-Conference on Artificial Neural Networks – Advances in Computational Intelligence, Lecture Notes in Computer Science, Springer, Cham, 2019, pp. 313–324. doi:10.1007/978-3-030-20518-8_27.

[65] W. Qiu, A. Yuille, UnrealCV: Connecting Computer Vision to Unreal Engine, in: G. Hua, H. Jégou (Eds.), European Conference on Computer Vision – Workshops, Vol. 9915, Springer, Cham, 2016, pp. 909–916. doi:10.1007/978-3-319-49409-8_75.

[66] J. Kang, B.-k. Jeon, S.-h. Kim, S.-y. Park, Exposition of Music: VR Exhibition, in: ACM SIGGRAPH Immersive Pavilion, New York, NY, USA, 2021, pp. 1–2. doi:10.1145/3450615.3464535.

[67] Z. Shen, J. Liu, Y. Zheng, L. Cao, A Low-cost Mobile VR Walkthrough System for Displaying Multimedia Works Based on Unity3D, in: 14th International Conference on Computer Science & Education (ICCSE), IEEE, Toronto, ON, Canada, 2019, pp. 415–419. doi:10.1109/iccse.2019.8845390.

[68] N. Statham, Use of Photogrammetry in Video Games: A Historical Overview, Games and Culture 15 (3) (2020) 289–307. doi:10.1177/1555412018786415.

[69] M. Ziegler, M. Bemana, J. Keinert, K. Myszkowski, Near Real-time Light Field Reconstruction and Rendering for On-Line Light Field Evaluation, in: European Light Field Imaging Workshop

(ELFI), Borovets, Bulgaria, 2019, p. 4.

[70] G. D. de Dinechin, A. Paljic, From Real to Virtual: An Image-Based Rendering Toolkit to Help Bring the World Around Us Into Virtual Reality, in: IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW), Atlanta, GA, USA, 2020, p. 6. doi:10.1109/vrw50115.2020.00076.

[71] Rendering Pipeline Overview - OpenGL Wiki, https://www.khronos.org/opengl/wiki/Rendering_Pipeline_Overview (Last accessed on 06.12.2021).

[72] R. S. Overbeck, D. Erickson, D. Evangelakos, M. Pharr, P. Debevec, A System for Acquiring, Processing, and Rendering Panoramic Light Field Stills for Virtual Reality, ACM Transactions on Graphics 37 (6) (2019) 1–15. arXiv:1810.08860, doi:10.1145/3272127.3275031.

[73] D. Yue, M. S. K. Gul, M. Bätz, J. Keinert, R. Mantiuk, A Benchmark of Light Field View Interpolation Methods, in: IEEE International Conference on Multimedia Expo Workshops (ICMEW), London, UK, 2020, pp. 1–6. doi:10.1109/icmew46912.2020.9106041.

[74] Y. Yoon, H.-G. Jeon, D. Yoo, J.-Y. Lee, I. S. Kweon, Light-Field Image Super-Resolution Using Convolutional Neural Network, IEEE Signal Processing Letters 24 (6) (2017) 848–852. doi:10.1109/lsp.2017.2669333.

[75] M. S. K. Gul, B. K. Gunturk, Spatial and Angular Resolution Enhancement of Light Fields Using Convolutional Neural Networks, IEEE Transactions on Image Processing 27 (5) (2018) 2146–2159. doi:10.1109/tip.2018.2794181.

[76] G. Wu, M. Zhao, L. Wang, Q. Dai, T. Chai, Y. Liu, Light Field Reconstruction Using Deep Convolutional Network on EPI, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 2017, pp. 1638–1646. doi:10.1109/cvpr.2017.178.

[77] Y. Wang, F. Liu, Z. Wang, G. Hou, Z. Sun, T. Tan, End-to-End View Synthesis for Light Field Imaging with Pseudo 4DCNN, in: V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss (Eds.), European Conference on Computer Vision (ECCV), Vol. 11206, Springer, Cham, 2018, pp. 340–355. doi:10.1007/978-3-030-01216-8_21.

[78] N. K. Kalantari, T.-C. Wang, R. Ramamoorthi, Learning-based view synthesis for light field cameras, ACM Transactions on Graphics 35 (6) (2016) 1–10. doi:10.1145/2980179.2980251.

[79] J. Jin, J. Hou, H. Yuan, S. Kwong, Learning Light Field Angular Super-Resolution via a Geometry-Aware Network, Proceedings of the AAAI Conference on Artificial Intelligence 34 (07) (2020) 11141–11148. doi:10.1609/aaai.v34i07.6771.

[80] S. Niklaus, L. Mai, F. Liu, Video Frame Interpolation via Adaptive Separable Convolution, in: IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 261–270. doi:10.1109/iccv.2017.37.

[81] P. P. Srinivasan, T. Wang, A. Sreelal, R. Ramamoorthi, R. Ng, Learning to Synthesize a 4D RGBD Light Field from a Single Image (arXiV), arXiv (Aug. 2017). arXiv:1708.03292.

[82] J. Navarro, N. Sabater, Learning Occlusion-Aware View Synthesis for Light Fields, arXiv (May 2019). arXiv:1905.11271.

[83] M. S. K. Gul, M. U. Mukati, M. Bätz, S. Forchhammer, J. Keinert, Light-Field View Synthesis Using A Convolutional Block Attention Module, in: IEEE International Conference on Image Processing (ICIP), Anchorage, AK, USA, 2021, pp. 3398–3402. doi:10.1109/icip42928.2021.9506586.

[84] S. Woo, J. Park, J.-Y. Lee, I. S. Kweon, CBAM: Convolutional Block Attention Module, in: European Conference on Computer Vision (ECCV), Springer, Cham, 2018, p. 17. doi:10.1007/978-3-030-01234-2_1.

[85] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, R. Ng, NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), European Conference on Computer Vision (ECCV), Springer, Cham, 2020, pp. 405–421.

[86] Thorsten Herfet, Kelvin Chelli, Mikael Le Pendu, Light field representation - The dimensions in light fields, in: Immersive Video Technologies, Academic Press, 2022, pp. 173–199.

[87] J. Thies, M. Zollhöfer, M. Nießner, Deferred neural rendering: Image synthesis using neural textures, ACM Transactions on Graphics 38 (4) (2019) 1–12. arXiv:1904.12356, doi:10.1145/3306346.3323035.

[88] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, S. Suwajanakorn, NeX: Real-Time View Synthesis With Neural Basis Expansion, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 8534–8543.

[89] L. Liu, J. Gu, K. Zaw Lin, T.-S. Chua, C. Theobalt, Neural Sparse Voxel Fields, in: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), Conference on Neural Information Processing Systems (NeurIPS), Vol. 33, Virtual, 2020, pp. 15651–15663.

[90] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, Y. Sheikh, Neural Volumes: Learning Dynamic Renderable Volumes from Images, ACM Transactions on Graphics 38 (4) (2019) 1–14. arXiv:1906.07751, doi:10.1145/3306346.3323020.

[91] Q. Wang, Z. Wang, K. Genova, P. P. Srinivasan, H. Zhou, J. T. Barron, R. Martin-Brualla, N. Snavely, T. Funkhouser, IBRNet: Learning Multi-View Image-Based Rendering, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 4690–4699.

[92] S. Lombardi, T. Simon, G. Schwartz, M. Zollhoefer, Y. Sheikh, J. Saragih, Mixture of Volumetric Primitives for Efficient Neural Rendering, ACM Transactions on Graphics 40 (4) (Jul. 2021). doi:10.1145/3450626.3459863.

[93] T. Neff, P. Stadlbauer, M. Parger, A. Kurz, J. H. Mueller, C. R. A. Chaitanya, A. Kaplanyan, M. Steinberger, DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks, Computer Graphics Forum 40 (4) (2021) 45–59. doi:10.1111/cgf.14340.

[94] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, G. Brostow, Deep blending for free-viewpoint image-based rendering, ACM Transactions on Graphics 37 (6) (2019) 1–15. doi:10.1145/3272127.3275084.

[95] C. Griwodz, S. Gasparini, L. Calvet, P. Gurdjos, F. Castan, B. Maujean, G. De Lillo, Y. Lanthony, AliceVision Meshroom: An open-source 3D reconstruction pipeline, in: MMSys: ACM Multimedia Systems Conference, Istanbul Turkey, 2021, pp. 241–247. doi:10.1145/3458305.3478443.

[96] RealityCapture: Mapping and 3D Modeling Photogrammetry Software - CapturingReality.com, https://www.capturingreality.com/ (Last accessed on 06.12.2021).

[97] J. L. Schönberger, E. Zheng, J.-M. Frahm, M. Pollefeys, Pixelwise View Selection for Unstructured Multi-View Stereo, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), European Conference on Computer Vision (ECCV), Vol. 9907, European Conference on Computer Vision (ECCV), Springer, Cham, 2016, pp. 501–518. doi:10.1007/978-3-319-46487-9_31.

[98] C. Reiser, S. Peng, Y. Liao, A. Geiger, KiloNeRF: Speeding Up Neural Radiance Fields With Thousands of Tiny MLPs, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 14335–14345.

[99] K. Zhang, G. Riegler, N. Snavely, V. Koltun, NeRF++: Analyzing and Improving Neural Radiance Fields, arXiv (Oct. 2020). arXiv:2010.07492.

[100] R. Martin-Brualla, N. Radwan, M. S. M. Sajjadi, J. T. Barron, A. Dosovitskiy, D. Duckworth, NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 7206–7215. doi:10.1109/cvpr46437.2021.00713.

[101] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, G. Wetzstein, Implicit Neural Representations with Periodic Activation Functions, in: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), Conference on Neural Information Processing Systems (NeurIPS), Vol. 33, Virtual, 2020, pp. 7462–7473.

[102] J. T. Barron, B. Mildenhall, M. Tancik, P. Hedman, R. Martin-Brualla, P. P. Srinivasan, Mip-NeRF: A Multiscale Representation for Anti-Aliasing Neural Radiance Fields, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5855–5864.

[103] A. Jain, M. Tancik, P. Abbeel, Putting NeRF on a Diet: Semantically Consistent Few-Shot View Synthesis, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5885–5894.

[104] J. N. Martel, D. B. Lindell, C. Z. Lin, E. R. Chan, M. Monteiro, G. Wetzstein, ACORN: Adaptive Coordinate Networks for Neural Representation, ACM Transactions on Graphics 40 (4) (2021). doi:10.1145/3450626.3459785.

[105] S. Peng, C. M. Jiang, Y. Liao, M. Niemeyer, M. Pollefeys, A. Geiger, Shape As Points: A Differentiable Poisson Solver, arXiv (Jun. 2021). arXiv:2106.03452.

[106] L. Yariv, J. Gu, Y. Kasten, Y. Lipman, Volume Rendering of Neural Implicit Surfaces, arXiv (Jun. 2021). arXiv:2106.12052.

[107] R. Arandjelović, A. Zisserman, NeRF in detail: Learning to sample for view synthesis, arXiv (Jun. 2021). arXiv:2106.05264.

[108] K. Deng, A. Liu, J.-Y. Zhu, D. Ramanan, Depth-supervised NeRF: Fewer Views and Faster Training for Free, arXiv (Jul. 2021). arXiv:2107.02791.

[109] H. Baatz, J. Granskog, M. Papas, F. Rousselle, J. Novák, NeRF-Tex: Neural Reflectance Field Textures, in: A. Bousseau, M. McGuire (Eds.), Eurographics Symposium on Rendering - DL-Only Track, The Eurographics Association, 2021, p. 13. doi:10.2312/sr.20211285.

[110] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, H. Su, MVSNeRF: Fast Generalizable Radiance Field Reconstruction From Multi-View Stereo, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 14124–14133.

[111] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, A. Kanazawa, PlenOctrees for Real-Time Rendering of Neural Radiance Fields, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5752–5761.

[112] S. Peng, M. Niemeyer, L. Mescheder, M. Pollefeys, A. Geiger, Convolutional Occupancy Networks, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), European Conference on Computer Vision (ECCV), Springer, Cham, 2020, pp. 523–540. doi:10.1007/978-3-030-58580-8_31.

[113] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, G. Wetzstein, MetaSDF: Meta-Learning Signed Distance Functions, in: Conference on Neural Information Processing Systems (NeurIPS), Virtual, 2020, pp. 10136–10147.

[114] J. Sun, Y. Xie, L. Chen, X. Zhou, H. Bao, NeuralRecon: Real-Time Coherent 3D Reconstruction From Monocular Video, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 15598–15607.

[115] J. Knodt, S.-H. Baek, F. Heide, Neural Ray-Tracing: Learning Surfaces and Reflectance for Relighting and View Synthesis, arXiv (Apr. 2021). arXiv:2104.13562.

[116] M. Oechsle, S. Peng, A. Geiger, UNISURF: Unifying Neural Implicit Surfaces and Radiance Fields for Multi-View Reconstruction, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5589–5599.

[117] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, P. Debevec, Baking Neural Radiance Fields for Real-Time View Synthesis, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5875–5884.

[118] D. Rebain, W. Jiang, S. Yazdani, K. Li, K. M. Yi, A. Tagliasacchi, DeRF: Decomposed Radiance Fields, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 14153–14161.

[119] P. Erler, P. Guerrero, S. Ohrhallinger, N. J. Mitra, M. Wimmer, Points2Surf Learning Implicit Surfaces from Point Clouds, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), European

Conference on Computer Vision (ECCV), Lecture Notes in Computer Science, Springer, Cham, 2020, pp. 108–124. doi:10.1007/978-3-030-58558-7_7.

[120] P. Wang, L. Liu, Y. Liu, C. Theobalt, T. Komura, W. Wang, NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction, arXiv (Jun. 2021). arXiv:2106.10689.

[121] B. Attal, E. Laidlaw, A. Gokaslan, C. Kim, C. Richardt, J. Tompkin, M. O'Toole, TöRF: Time-of-Flight Radiance Fields for Dynamic Scene View Synthesis, arXiv (Sep. 2021). arXiv:2109.15271.

[122] J. Li, Z. Feng, Q. She, H. Ding, C. Wang, G. H. Lee, MINE: Towards Continuous Depth MPI With NeRF for Novel View Synthesis, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 12578–12588.

[123] C. Chivriga, B. Wiberg, Y. Shentu, M. Loser, BUFF - Bounding unstructured radiance volumes for free view synthesis, https://github.com/qway/nerfmeshes (Last accessed on 06.12.2021).

[124] A. Yu, V. Ye, M. Tancik, A. Kanazawa, pixelNeRF: Neural Radiance Fields From One or Few Images, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 4578–4587.

[125] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, J. Valentin, FastNeRF: High-Fidelity Neural Rendering at 200FPS, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 14346–14355.

[126] Y. Liu, S. Peng, L. Liu, Q. Wang, P. Wang, C. Theobalt, X. Zhou, W. Wang, Neural Rays for Occlusion-aware Image-based Rendering, arXiv (Jul. 2021). arXiv:2107.13421.

[127] J. Shen, A. Ruiz, A. Agudo, F. Moreno, Stochastic Neural Radiance Fields: Quantifying Uncertainty in Implicit 3D Representations, arXiv (Sep. 2021). arXiv:2109.02123.

[128] Y. Wei, S. Liu, Y. Rao, W. Zhao, J. Lu, J. Zhou, NerfingMVS: Guided Optimization of Neural Radiance Fields for Indoor Multi-View Stereo, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5610–5619.

[129] F. Cole, K. Genova, A. Sud, D. Vlasic, Z. Zhang, Differentiable Surface Rendering via Non-Differentiable Sampling, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 6088–6097.

[130] G.-W. Yang, W.-Y. Zhou, H.-Y. Peng, D. Liang, T.-J. Mu, S.-M. Hu, Recursive-NeRF: An Efficient and Dynamically Growing NeRF, arXiv (May 2021). arXiv:2105.09103.

[131] C.-H. Lin, W.-C. Ma, A. Torralba, S. Lucey, BARF: Bundle-Adjusting Neural Radiance Fields, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5741–5751.

[132] Z. Wang, S. Wu, W. Xie, M. Chen, V. A. Prisacariu, NeRF–: Neural Radiance Fields Without Known Camera Parameters, arXiv (Feb. 2021). arXiv:2102.07064.

[133] Y. Jeong, S. Ahn, C. Choy, A. Anandkumar, M. Cho, J. Park, Self-Calibrating Neural Radiance Fields, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5846–5854.

[134] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, R. Martin-Brualla, Nerfies: Deformable Neural Radiance Fields, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 5865–5874.

[135] K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, S. M. Seitz, HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields, arXiv (Jun. 2021). arXiv:2106.13228.

[136] A. Pumarola, E. Corona, G. Pons-Moll, F. Moreno-Noguer, D-NeRF: Neural Radiance Fields for Dynamic Scenes, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 10318–10327.

[137] Z. Li, S. Niklaus, N. Snavely, O. Wang, Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 6498–6508.

[138] P. P. Srinivasan, B. Deng, X. Zhang, M. Tancik, B. Mildenhall, J. T. Barron, NeRV: Neural Reflectance and Visibility Fields for Relighting and View Synthesis, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 7495–7504.

[139] M. Boss, R. Braun, V. Jampani, J. T. Barron, C. Liu, H. P. Lensch, NeRD: Neural Reflectance Decomposition From Image Collections, in: IEEE/CVF International Conference on Computer Vision (ICCV), Virtual, 2021, pp. 12684–12694.

[140] X. Zhang, P. P. Srinivasan, B. Deng, P. Debevec, W. T. Freeman, J. T. Barron, NeRFactor: Neural Factorization of Shape and Reflectance Under an Unknown Illumination, arXiv (Jun. 2021). arXiv:2106.01970.

[141] J. T. Barron, B. Mildenhall, D. Verbin, P. P. Srinivasan, P. Hedman, Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields, arXiv (Nov. 2021). arXiv:2111.12077.

[142] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, Q. Zhang, JAX: Composable transformations of Python+NumPy programs, http://github.com/google/jax (2018).

[143] B. Deng, J. T. Barron, P. P. Srinivasan, JaxNeRF: An efficient JAX implementation of NeRF, https://github.com/google-research/google-research/tree/master/jaxnerf (2020).

[144] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, R. Ng, Learned Initializations for Optimizing Coordinate-Based Neural Representations, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 2845–2854. doi:10.1109/cvpr46437.2021.00287.

[145] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009 IEEE conference on computer vision and pattern recognition, Miami, FL, USA, 2009, pp. 248–255. doi:10.1109/cvpr.2009.5206848.

[146] Alex Evans, Learning from failure: A Survey of Promising, Unconventional and Mostly Abandoned Renderers for 'Dreams PS4', a Geometrically Dense, Painterly UGC Game (Presentation), https://www.mediamolecule.com/blog/article/siggraph_2015 (2015).

[147] Sebastien Hillaire, Physically Based and Unified Volumetric Rendering in Frostbite (Presentation), https://www.slideshare.net/DICEStudio/physically-based-and-unified-volumetric-rendering-in-frostbite (2015).

[148] W. E. Lorensen, H. E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, ACM SIGGRAPH Computer Graphics 21 (4) (1987) 163–169. doi:10.1145/37402.37422.

[149] Signed distance function, Wikipedia (Last accessed on 06.12.2021).

[150] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, Y. Lipman, Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance, in: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), Conference on Neural Information Processing Systems (NeurIPS), Virtual, 2020, pp. 2492–2502.

[151] F. Xiang, Z. Xu, M. Hasan, Y. Hold-Geoffroy, K. Sunkavalli, H. Su, NeuTex: Neural Texture Mapping for Volumetric Neural Rendering, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Virtual, 2021, pp. 7119–7128.

[152] J. Hasselgren, J. Munkberg, J. Lehtinen, M. Aittala, S. Laine, Appearance-Driven Automatic 3D Model Simplification, arXiv (Apr. 2021). arXiv:2104.03989.

[153] M. Deering, S. Winner, B. Schediwy, C. Duffy, N. Hunt, The triangle processor and normal vector shader: A VLSI system for high performance graphics, ACM SIGGRAPH Computer Graphics 22 (4) (1988) 21–30. doi:10.1145/378456.378468.

[154] Deferred shading, Wikipedia (Last accessed on 06.12.2021).

[155] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: N. Navab, J. Hornegger, W. Wells, A. Frangi (Eds.), International Conference on Medical

Image Computing and Computer-Assisted Intervention (MICCAI), Springer, Springer, Cham, 2015, pp. 234–241. doi:10.1007/978-3-319-24574-4_28.

[156] G. Riegler, V. Koltun, Stable View Synthesis, arXiv (May 2021). arXiv:2011.07233.

[157] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, V. Lempitsky, Neural point-based graphics, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), European Conference on Computer Vision (ECCV), Lecture Notes in Computer Science, Springer, Cham, 2020, pp. 696–712. doi:10.1007/978-3-030-58542-6_42.

[158] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, R. Martin-Brualla, Neural rerendering in the wild, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 6878–6887. doi:10.1109/CVPR.2019.00704.

[159] D. Rückert, L. Franke, M. Stamminger, ADOP: Approximate Differentiable One-Pixel Point Rendering, arXiv (Oct. 2021). arXiv:2110.06635.

[160] A. W. Bergman, P. Kellnhofer, G. Wetzstein, Fast Training of Neural Lumigraph Representations using Meta Learning, arXiv (Jun. 2021). arXiv:2106.14942.

[161] A. Meka, R. Pandey, C. Häne, S. Orts-Escolano, P. Barnum, P. David-Son, D. Erickson, Y. Zhang, J. Taylor, S. Bouaziz, C. Legendre, W.-C. Ma, R. Overbeck, T. Beeler, P. Debevec, S. Izadi, C. Theobalt, C. Rhemann, S. Fanello, Deep relightable textures: Volumetric performance capture with neural rendering, ACM Transactions on Graphics 39 (6) (2020) 259:1–259:21. doi:10.1145/3414685.3417814.

[162] M. Oechsle, M. Niemeyer, C. Reiser, L. Mescheder, T. Strauss, A. Geiger, Learning implicit surface light fields, in: International Conference on 3D Vision (3DV), IEEE, Fukuoka, Japan, 2020, pp. 452–462. doi:10.1109/3dv50981.2020.00055.

[163] J. Thies, M. Elgharib, A. Tewari, C. Theobalt, M. Nießner, Neural voice puppetry: Audio-driven facial reenactment, in: A. Vedaldi, H. Bischof, T. Brox, J.-M. Frahm (Eds.), European Conference on Computer Vision (ECCV), Lecture Notes in Computer Science, Springer, Cham, 2020, pp. 716–731. doi:10.1007/978-3-030-58517-4_42.