Parallax360: Stereoscopic 360° Scene Representation for Head-Motion Parallax

Bicheng Luo, Feng Xu, Christian Richardt and Jun-Hai Yong

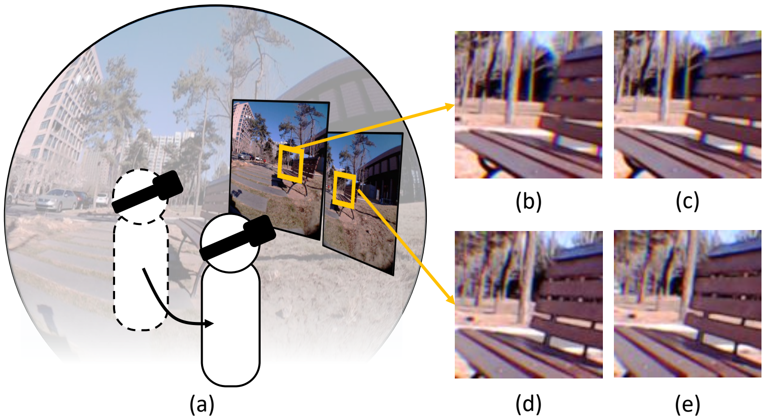

Fig. 1. We propose a novel image-based representation to generate novel views with head-motion parallax for a 360° scene: **(a)** When a viewer changes the orientation and position, correct novel views are presented. **(b, c)** The original stereo views. **(d, e)** The novel stereo views. Notice that the positional relationship between the bench and the trees changes according to the head motion.

Abstract—We propose a novel 360° scene representation for converting real scenes into stereoscopic 3D virtual reality content with head-motion parallax. Our image-based scene representation enables efficient synthesis of novel views with six degrees-of-freedom (6-DoF) by fusing motion fields at two scales: (1) disparity motion fields carry implicit depth information and are robustly estimated from multiple laterally displaced auxiliary viewpoints, and (2) pairwise motion fields enable real-time flow-based blending, which improves the visual fidelity of results by minimizing ghosting and view transition artifacts. Based on our scene representation, we present an end-to-end system that captures real scenes with a robotic camera arm, processes the recorded data, and finally renders the scene in a head-mounted display in real time (more than 40 Hz). Our approach is the first to support head-motion parallax when viewing real 360° scenes. We demonstrate compelling results that illustrate the enhanced visual experience – and hence sense of immersion – achieved with our approach compared to widely-used stereoscopic panoramas.

Index Terms—360° scene capture, scene representation, head-motion parallax, 6 degrees-of-freedom (6-DoF), image-based rendering

1 Introduction

Experiencing virtual reality (VR) has become increasingly easy in recent years thanks to the availability of commodity head-mounted displays (HMDs). However, good VR content is still scarce and might become a bottleneck for VR technology. Currently, most VR content is computer-generated from virtual 3D scenes, which are tedious to model and animate, and therefore often lack the realism of real scenes. On the other hand, capturing real scenes and converting them into highly realistic VR content is a promising avenue for creating visually rich VR content. Many applications, such as virtual tourism, teleconferencing, film and television production would benefit from such VR content that is captured from the real world.

The most common representations for 360° VR content of real scenes are panoramas (360° images) and stereo panoramas. Panoramas can be easily obtained by stitching multiple images [41], but they do not carry any 3D information of the scene. Stereo panoramas, on the other hand, contain parallax (or disparity) [31,39]. However, as all the scene information is represented by two panoramas with a fixed parallax, head-motion parallax – where the occlusion relationship changes with the viewpoint – cannot be generated by stereo panoramas, as it requires a different parallax corresponding to different head positions. One potential solution for full 3D perception for arbitrary viewpoints is to reconstruct the complete 3D geometry of the scene. However, this is very difficult to achieve for arbitrary real scenes, as real scenes are generally large and complex, while the viewpoints for recording are usually restricted to a small region.

We propose a novel image-based representation for modeling a 3D 360° scene using implicit depth information (see Figure 1). The core of our representation builds on two-scale motion fields, disparity and pairwise motion fields, which extract 3D information of the scene and improve the rendering quality and performance. Using our representation, we do not require a dense light-field sampling with huge recording and storage cost, but only 72×8 images with the two-scale motion fields. In addition, our representation enables real-time viewpoint synthesis and smooth viewpoint transitions without requiring explicit 3D scene geometry. Our approach makes the following contributions:

- A novel image-based representation for 360° real scenes. Based on this representation, we propose an end-to-end system that

- *Bicheng Luo, Feng Xu (corresponding author) and Jun-Hai Yong are with the School of Software at Tsinghua University, China. E-mail: luobc14@mails.tsinghua.edu.cn, feng-xu@tsinghua.edu.cn, yongjh@tsinghua.edu.cn.*
- *Christian Richardt is with the University of Bath, UK. E-mail: christian@richardt.name.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

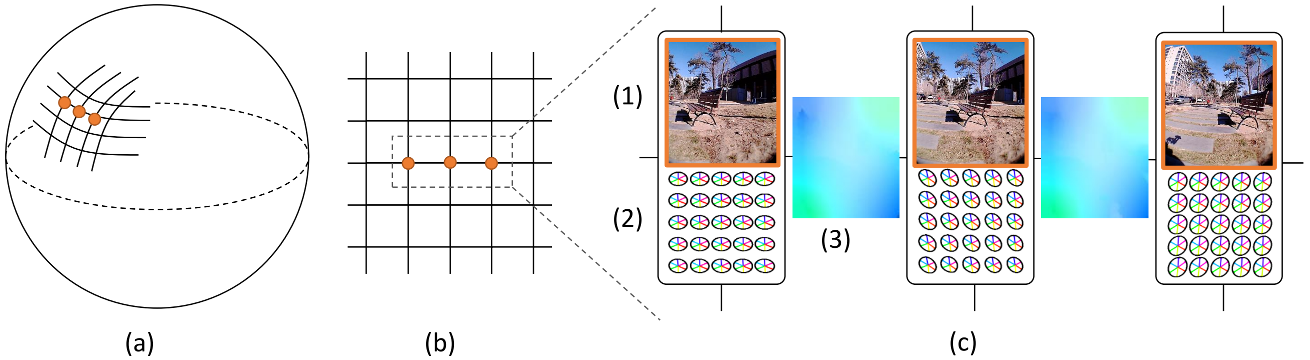

Fig. 2. Illustration of our 360° scene representation: **(a)** Key frames are captured uniformly on the surface of a sphere (orange circles). **(b)** The local region on the sphere is remapped into the 2D latitude-longitude plane. **(c)** The three types of information stored in our representation: (1) key frames (bordered in orange), (2) disparity motion fields (the curves fitted to the colored motion vectors), and (3) pairwise motion fields (the color-coded flow fields between adjacent key frames).

ranges from recording to rendering of real scenes. The results show that our system enables head-motion parallax, which is missing from existing panorama-based approaches.

- A robust curve-based fitting method for estimating disparity motion fields, which implicitly conveys depth information for a viewpoint. Disparity motion fields are estimated more robustly compared to traditional stereo matching-based depth estimation.
- Novel-view synthesis with real-time flow-based blending between synthesized images from different viewpoints produces smooth viewpoint transitions with minimal visual artifacts. By combining disparity and pairwise motion fields on the fly, we avoid costly online motion estimation and achieve real-time rendering.

2 Related Work

Image-Based Scene Modeling A 3D scene can be modeled with different forms of information,and visualized with different image-based rendering techniques [37]. On the one hand, scenes can be represented purely with images and without any geometric information, using a light field [24] or lumigraph [13]. Even though techniques in this category are highly developed [23,25,27,36,47], they still require extremely densely sampled images of a scene, which is impractical for a 360° scene. To more efficiently represent a scene, image-based rendering uses sparsely sampled images with different forms of depth information, such as the unstructured lumigraph [5] that generalizes light-field techniques. Using depth for synthesizing novel viewpoints has been shown to produce increasingly high-quality results [6,7,10,12,15,42]. However, as these depth-based techniques are not designed for 360° scenes, it is unclear how to sample an entire 360° scene to guarantee both a small baseline between images (which is desirable for view synthesis) and an efficient representation with a low sampling density of images. Huang et al. [16] use dense scene reconstruction to create VR videos with head-motion parallax from a single 360° video. However, geometry-based image warping limits the visual quality of results, as straight lines are bending, and occlusions lead to stretching artifacts, as the viewer moves away from the capture viewpoint.

3D Scene Reconstruction On the other hand, a virtual 3D scene can be very efficiently rendered if it is available in the form of textured geometry. PMVS [11] is widely used for reconstructing 3D models of real scenes from multi-view input images. Using depth sensors, larger, mostly indoor, scenes can also be reconstructed [2,29,30,43,46], but with a relatively long capturing process. Besides static scenes, dynamic objects are also able to be reconstructed with multiview input [8,9] or single-view input [18,28]. Hedman et al. [14] reconstruct textured geometry from casually captured input photos, but their approach cannot handle view-dependent effects. However, current 3D reconstruction techniques are still not able to handle large outdoor scenes, due to the lack of ability to reconstruct distant objects and the difficulty of achieving multi-view coverage of all objects in a scene.

Panoramas Currently, the most convenient way to visualize a real 360° scene within a head-mounted display is to use a panorama (or 360° video), which is created by aligning and stitching multiple images (or videos) corresponding to different viewing directions. Panorama stitching techniques are widely covered in the literature [4,33,41]. For generating panoramas, Zelnik-Manor et al. [45] project the images onto a sphere to represent the 360° scene. Kopf et al. [20] propose locally-adapted projections to reduce distortions in the generated panoramas. Optical flow is also used to minimize undesired motion parallax in synthesizing panoramas [40], as it compensates motions in the scene. Also based on optical flow, Kang et al. [19] propose multi-perspective plane sweeping to seamlessly stitch images. Perazzi et al. [32] stitch panoramic videos from unstructured camera arrays by removing parallax with local warps. We also use flow-based blending to seamlessly align and fuse multiple synthesized images into a single coherent image. Recently, Lee et al. [22] used a deformed sphere to perform spherical projection, and a non-uniform sampling to represent important regions in a scene with higher resolution.

Stereo Panoramas Stereo panoramas are a better representation for 360° scenes than a single panorama, as the latter do not provide any 3D information. Stereo panoramas can be generated by moving a camera along a circular trajectory with radial [31] or tangential [38] viewing direction, and stitching vertical stripes from the images. At the same time, the depth of a panorama can also be estimated, and thus stereo vision can be achieved [39]. Recently, Richardt et al. [35] proposed a practical solution for creating high-quality stereo panoramas by stabilizing and correcting input images, and seamless stitching them using flow-based ray upsampling. Custom multi-camera rigs also enable capture of stereo video panorama to represent dynamic scenes [1,26]. However, techniques based on stereo panoramas assume that the viewer only rotates around a vertical axis between the two eyes, and thus stereo vision is limited to a fixed viewpoint with varying view directions, but without head-motion parallax.

Light fields have also been used for stitching panoramas [3] and estimating depth maps [21] of 360° scenes. Our 360° scene representation overcomes these limitations by using image-based rendering with multiple key frames, leading to results of higher visual quality.

3 Parallax360 Scene Representation

Our 360° scene representation consists of three types of information (illustrated in Figure 2):

1. **key frames** that represent the color information of the scene,
2. **disparity motion fields** that represent implicit 3D information of the scene at each key frame, and
3. **pairwise motion fields** for efficient and smooth viewpoint transitions in novel-view synthesis.

Disparity and pairwise motion fields form our two-scale motion fields.

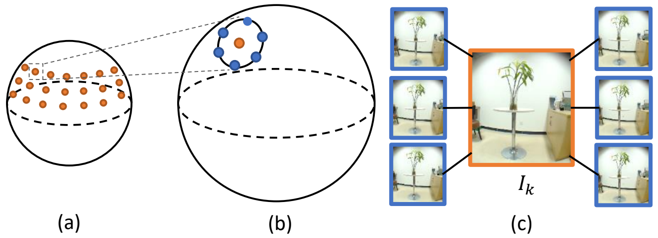

Fig. 3. Our capture scheme: **(a)** Key frames on the sampling sphere, as orange points. **(b)** The sampling circle of relative frames (blue points) around one key frame. **(c)** One key frame I_k (bordered in orange), and its six relative frames (bordered in blue).

Key Frames To capture a complete scene in 360°, we sample discrete positions uniformly in latitude and longitude on a sphere (called the 'sampling sphere'), and record key frame images at these positions looking radially outward (see Figure 2a,b). In all our experiments, we capture key frames at uniform angular increments of 5° in both latitude and longitude on a sphere with a diameter of 1.25 m.

Disparity Motion Fields While a key frame describes the visual information of one specific viewpoint, the disparity motion field conveys its implicit depth information. Specifically, the disparity represents the motion between a point and its corresponding points in images from surrounding viewpoints. For this, we extend the common case of binocular disparity in stereo matching to multiple viewpoints surrounding one key frame. We estimate the motion vectors for six surrounding images for each key frame, and represent the motion vectors using curves, as illustrated in Figure 2c, for more robust motion estimation and efficient computation during view synthesis. Given the key frames, the curve-based disparity motion fields can be used to synthesize nearby viewpoints, similar to depth information used for view synthesis.

Our curve-based representation has three advantages over explicit depth maps. First, it does not require camera calibration which is normally necessary for depth estimation with stereo matching. Second, the curve is fitted to multiple motion vectors, and is thus more robust to errors in the motion estimation. Third, our representation is redundant compared with explicit depth maps, which is useful for synthesizing novel viewpoints in different directions. We discuss the computation of disparity motion fields in Section 4.2, and their usage in Section 4.3.1.

Pairwise Motion Fields Given a key frame and its disparity motion field, we can render novel viewpoints near the position of the key frame. However, if the viewpoint moves from one key frame to another, the sudden switch between key frames will generate noticeable popping artifacts. To solve this problem, and achieve a smooth transition between key frames, the flow fields between their respective results are required to synthesize a smooth interpolation. In our approach, we precompute the optical flow between pairs of adjacent key frames, and calculate the blending motion fields online using simple flow arithmetic. This avoids expensive online optical flow estimation, and ensures real-time performance. The pairwise motion fields are visualized in Figure 2c, and we describe their usage in Section 4.3.

4 Method

The pipeline of our end-to-end system comprises three steps: image capture, motion field precomputation and novel-view synthesis.

4.1 360° Image Capture

Besides the key frames, we also capture another kind of frames, called *relative frames*, which we use to compute the disparity motion fields. Specifically, for a key frame I_k, we first capture it at its defined position on the sampling sphere with radially outward viewing direction. We then capture six relative frames at positions close to the key frame. In our approach, relative frames are captured on a circle centered at the key frame position, with a fixed radius of 25 mm in our experiments. Figure 3b shows the sampling pattern of relative frames in our approach. Finally, these seven images (shown in Figure 3c) are grouped together for the next steps. Note that the relative frames are discarded after the motion field computation, and only the key frames are kept.

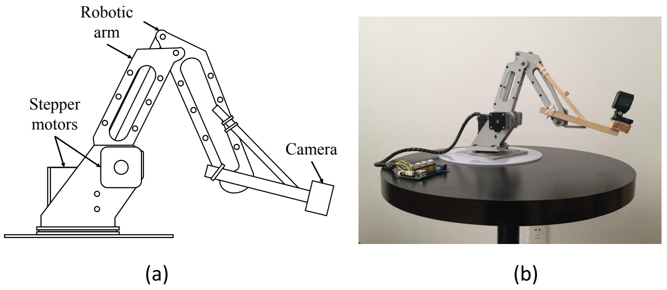

Fig. 4. Our robotic capture device for real scenes: **(a)** Schematic drawing with the main components, and **(b)** photo of our capture device.

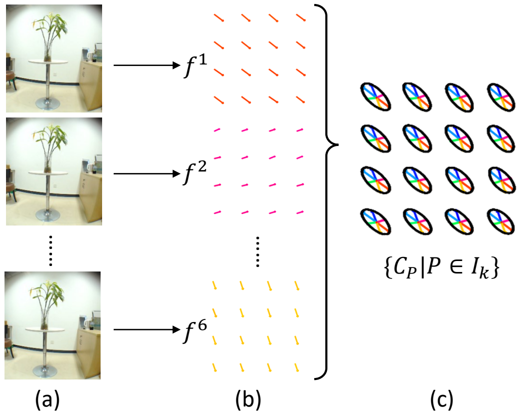

Fig. 5. Disparity motion fields: **(a)** Relative frames. **(b)** Motion fields $\{f^1, f^2, \ldots, f^6\}$ between the key frame and each relative frame. **(c)** Disparity motion curves $\{C_P\}$ for each image patch P in the key frame I_k.

To capture the key frames and the corresponding relative frames for a real scene, we developed the capture device in Figure 4. We use a robotic arm with three degrees-of-freedom to control the position and orientation of a camera. We convert sampling coordinates to the movement of the stepper motors, so that the camera can be placed at the camera position required by our capture scheme. In our experiments, we capture a range of 360° horizontally and 40° vertically in steps of 5°. Our device captures the corresponding $72 \times 8 = 576$ key frames and $576 \times 6 = 3{,}456$ relative frames in less than 2 hours. This time can be reduced by using faster stepper motors.

4.2 Motion Field Precomputation

In this section, we introduce the computation of the disparity motion fields from the captured key frames and relative frames, and then discuss computation of the pairwise motion fields between key frames.

4.2.1 Disparity Motion Fields

Inspired by the motion compensation techniques used in video compression, we use motion fields to synthesize new views, rather than rely on an estimate of scene depth, which is more difficult to estimate accurately, particularly for large outdoor scenes. The disparity motion field defines the 2D flow field from a key frame to any viewpoint surrounding the key frame, and is used to synthesize these novel views. All disparity motion fields are computed independently.

We start by computing optical flow [44] between the key frame and all its relative frames. Let us denote the flow fields using $F = \{f^1, f^2, \ldots, f^6\}$ (see Figure 5b), where $f^i(p)$ is the motion vector for pixel p in the flow field f^i. For storage and computational efficiency, we aggregate the motion vectors of individual pixels p into non-overlapping image patches P of size 8×8 pixels by averaging them. As

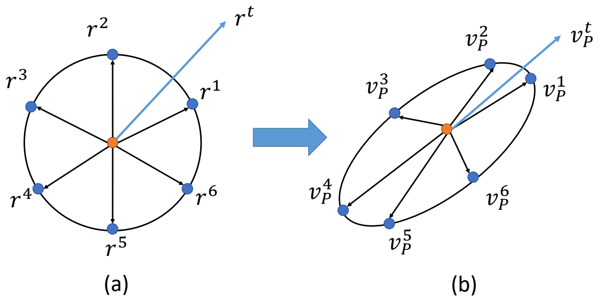

Fig. 6. Motion field interpolation using the disparity motion fields: **(a)** The sampling circle with six relative frames r^i and the target viewpoint r^t. **(b)** The motion vectors of the patch P in the six relative frames (v_P^i for $i = 1, \ldots, 6$), and in the target frame t (v_P^t).

demonstrated by our results, patch-level motion fields are sufficient for synthesizing high-quality novel views.

To merge the individual motion vectors of the relative frames into a single disparity motion field, we propose a curve-based motion representation. While the sampling pattern of relative frames is regular by construction (see Figure 6a), the motion vectors v_P^i corresponding to the relative frames are generally irregular (see Figure 6b). To achieve a robust motion encoding, we therefore separately encode the magnitude and direction of motion vectors. We fit an ellipse C_P (by least-squares fitting [34]) to the endpoints of all six motion vectors to encode the motion magnitude, and separately store the polar angles θ_P^i of the motion vectors v_P^i to encode their direction.

The parameters for the ellipses C_P and motion directions θ_P^i define our disparity motion fields, which are stored in our final scene representation. Notice that the fitted curves provide a robust fit to the six input motion vectors. The estimation errors in individual vectors are reduced by the fitting. After computing the disparity motion field (e.g. visualized in Figure 5c) for each key frame, we discard all relative frames as they are not required any more. As we will see in Section 4.3.1, this representation is well-suited for efficiently interpolating motion fields for any novel viewpoint near the key frame.

4.2.2 Pairwise Motion Fields

We compute the pairwise motion fields $f_{i \to j}$ between every pair I_i and I_j of neighboring key frames using optical flow [44]. For storage efficiency, we then again downsample the pairwise motion fields to the same resolution as the disparity motion fields, i.e. to one motion vector per patch of 8×8 pixels. These pairwise motion fields are also stored in our scene representation, and make it complete.

4.3 Novel-View Synthesis

We synthesize novel viewpoints anywhere inside the sampling sphere using a two-step process: (1) we synthesize a novel viewpoint on the sampling sphere, with a radially-outward viewing direction (Section 4.3.2), and then (2) we warp the synthesized image to match the desired viewpoint in the sphere, with any viewing direction (Section 4.3.3). However, before discussing these steps, we first describe our approach for interpolating disparity motion fields for a novel viewpoint (Section 4.3.1), which is used in the first step.

4.3.1 Disparity Motion Field Interpolation

We propose a coordinates transfer scheme to interpolate the disparity motion field to obtain the motion field for any novel viewpoint (tangent to the sampling sphere). First, based on the sampling scheme introduced in Section 4.1 and Figure 3, we represent the sampling positions of a key frame and its relative frames as the center of a circle and points on the circle (see Figure 6a). The vectors r^i indicate their displacement relative to the key frame. The position of any novel target viewpoint r^t can be represented similarly.

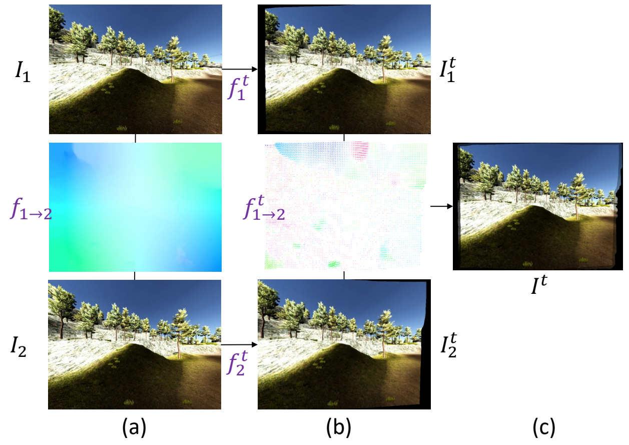

Fig. 7. The process of synthesizing a novel view (from two key frames): **(a)** The inputs are two key frames, I_1 and I_2, and their pairwise motion field $f_{1\to2}$. **(b)** The two intermediate interpolated images I_1^t and I_2^t, and the blending flow field $f_{1\to2}^t$ between them. **(c)** The final target view I^t.

We next express the direction of the novel viewpoint r^t as a linear combination of the nearest two relative frames, e.g. r^1 and r^2 in Figure 6a, using $\theta(r^t) = \alpha\theta(r^1) + \beta\theta(r^2)$ in this case. Here $\alpha + \beta = 1$ and $\theta(r)$ denotes the polar angle of the 2D vector r. Then, we use the coefficients α and β to interpolate the target motion field for this novel viewpoint using the curve-based motion field C_P and θ_P^i (see Figure 6b). To be specific, for a patch P in the key frame, we compute its motion vector using $v_P^t = \frac{\|r^t\|}{\|r^1\|} C_P(\alpha \cdot \theta_P^1 + \beta \cdot \theta_P^2)$. Here $C_P(\theta)$ is a motion vector from the center to the point on the ellipse with the polar angle of θ. We obtain the target motion field for the novel viewpoint by estimating the motion vectors for all patches accordingly.

4.3.2 Novel-View Synthesis on the Sampling Sphere

For a target viewpoint on the sampling sphere, we first find its K nearest key frames, ordered by distance (from nearest to furthest), which we denote using $I_1, I_2, \ldots, I_K$ without loss of generality. Each of the key frames is used to synthesize a target image I_k^t via their disparity motion field, and these target images are then aligned and fused into the final target image I^t via flow-based blending with the pairwise motion fields.

First, using the disparity motion field of key frame k, we obtain the motion field f_k^t between I_k and the target image I_k^t using the interpolation method in Section 4.3.1. Next, we use this disparity-derived motion field f_k^t to synthesize the target view I_k^t using

$$I_k^t(p) = I_k\Big((f_k^t)^{-1}(p)\Big), \tag{1}$$

where p is a pixel in the target image I_k^t, and $(f_k^t)^{-1}$ transforms the pixels of the target frame I_k^t to the key frame I_k. Notice that the motion field f_k^t is originally defined per patch P, and not per pixel p. We thus use bilinear interpolation on the 2D image domain to smoothly propagate the patch-level motion field to all pixels of the image. Figure 7ab shows an example for two key frames, I_1 and I_2, and the target views I_1^t and I_2^t interpolated from them.

If we simply alpha-blended all interpolated target views $I_1^t, I_2^t, \ldots, I_K^t$, the final output I^t would likely contain ghosting artifacts, because there is no guarantee that the same pixel on the individually interpolated target views $I_1^t, I_2^t, \ldots, I_K^t$ corresponds to the same scene point. If we set $K = 1$ (using only the nearest key frame), there will be no ghosting artifacts, but the viewpoint transition will not be smooth when the used key frames switches. This is known as ‘popping’ artifacts.

To synthesize a final target image I^t without ghosting artifacts, we align the target images of all key frames using flow-based blending. We estimate the blending motion field $f_{1\to k}^t$ using optical flow from target image I_1^t to target image I_k^t, as shown in Figure 7b. Ideally, the

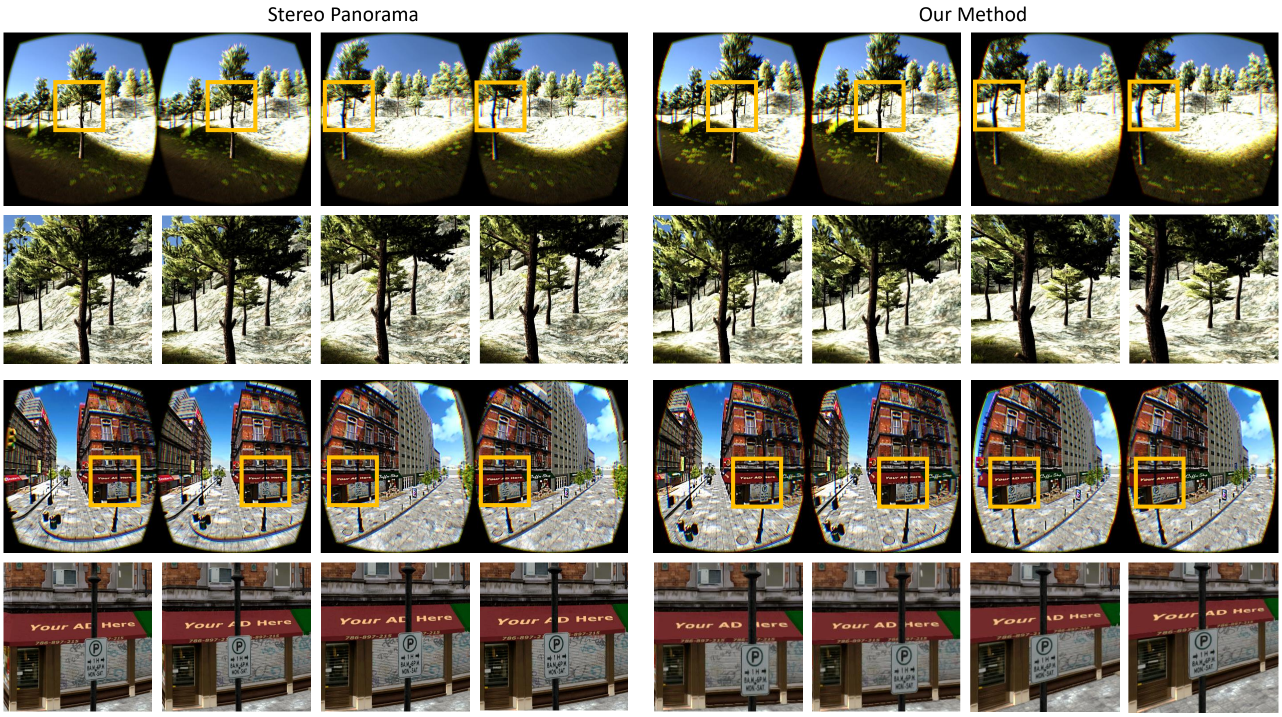

Fig. 8. Comparison of our method (right) to stereo panoramas [35] (left) on two synthetic scenes. For each method, we show two stereo pairs viewed from different viewpoints (left/right). Zoomed crops of each view are shown below each result. Our method preserves head-motion parallax between different viewpoints, as seen in the displacement of the nearby tree (top) or lamp post (bottom) compared to the trees (top) or shop window (bottom) in the background. Note that the full views of these results are captured directly from the DK2 headset, which has a lower resolution than the input images, and applies chromatic aberration compensation for the headset's optics, resulting in shifted color channels and thus color fringes.

pixel $p_k = f^t_{1\to k}(p_1)$ in I^t_k should correspond to the same scene point as pixel p_1 in I^t_1. So, given all the blending motion fields $f^t_{1\to k}$, we get the corresponding pixel coordinates of the scene point in all the images I^t_k. Then we can estimate its final pixel position in I_t by weighted averaging, where the weight is inversely proportional to the distance r^t_k between the target viewpoint and key frame k in their sampling circle spaces:

$$w_k = 1 - \frac{r^t_k}{\sum_{i=1}^{K} r^t_i}, \quad \text{or} \quad \overline{w}_k = \frac{w_k}{\sum_{i=1}^{K} w_i} = \frac{w_k}{K-1} \tag{2}$$

when the weights are normalized to sum to one. Mathematically, we then fuse the position of the corresponding pixels using

$$p = \sum_{k=1}^{K} \overline{w}_k \cdot p_k, \tag{3}$$

where p is the final position of the point in the fused target image I^t. Next, we calculate the color of the point p by fusing the colors of the corresponding points in the synthesized images $\{I^t_k \mid k = 1, \ldots, K\}$:

$$I^t(p) = \sum_{k=1}^{K} \overline{w}_k \cdot I^t_k(p_k). \tag{4}$$

Notice that p may fall between the pixel grid on image I^t; we again use bilinear interpolation on the 2D image plane to estimate the coordinates of all p_ks for points p at the center of a pixel grid.

Another obstacle that prevents achieving real-time performance is the calculation of the blending flow fields $f^t_{i\to k}$. We propose to use the precomputed disparity and pairwise motion fields (Section 4.2) to infer $f^t_{i\to k}$, rather than calculate optical flow between the target images I^t_i and I^t_k online. Starting from the target image I^t_i, we first apply the inverse of the disparity-derived motion field f^t_i, to map pixels into the coordinate frame of key frame I_i. We then apply the pairwise motion field $f_{i\to k}$ to map to key frame I_k. Finally, we apply the disparity-derived motion field f^t_k, to arrive in the coordinates of the target image I^t_k. Mathematically, we can express this using the flow concatenation operator '$\circ$' as

$$f^t_{i\to k} = f^t_k \circ f_{i\to k} \circ (f^t_i)^{-1}. \tag{5}$$

The whole synthesis process is illustrated for two key frames in Figure 7. As flow concatenation is essentially addition, the blending motion fields $f^t_{i\to k}$ can now be computed very efficiently.

To obtain stereoscopic results, we render two separate views, one for each eye of the head-mounted display.

In our experiments, we use the nearest 2–3 key frames (i.e. $K = 2, 3$), which strikes a suitable balance between synthesis quality and performance. To achieve real-time performance, the algorithm in this section is implemented on the GPU using Direct3D 11 HLSL. We compare the performance of CPU and GPU implementations in Section 5.

4.3.3 Viewpoint Extension

For a target viewpoint inside the sampling sphere, we first find the intersection of the target viewing ray and the sampling sphere. We then synthesize the target image at this intersection point, looking radially outward, following the method described in the previous section. Finally, to match the target view, we apply a homography warp that first rotates the virtual camera from the synthesized to the target viewing direction on the sampling sphere, and then applies a scaling transform that approximates the change in viewpoint from the position on the sphere to the target viewpoint inside the sphere. Although this synthesis scheme does not exactly reflect the change in perspective resulting from moving the viewpoint inside the sampling sphere, we find that, in practice, it is an extremely efficient approximation that nonetheless produces visually highly compelling results, as demonstrated in the following section. We restrict the distance between the target viewpoint and the sampling sphere to be less than a tenth of the diameter of the sampling sphere. Otherwise, artifacts may appear in the results.

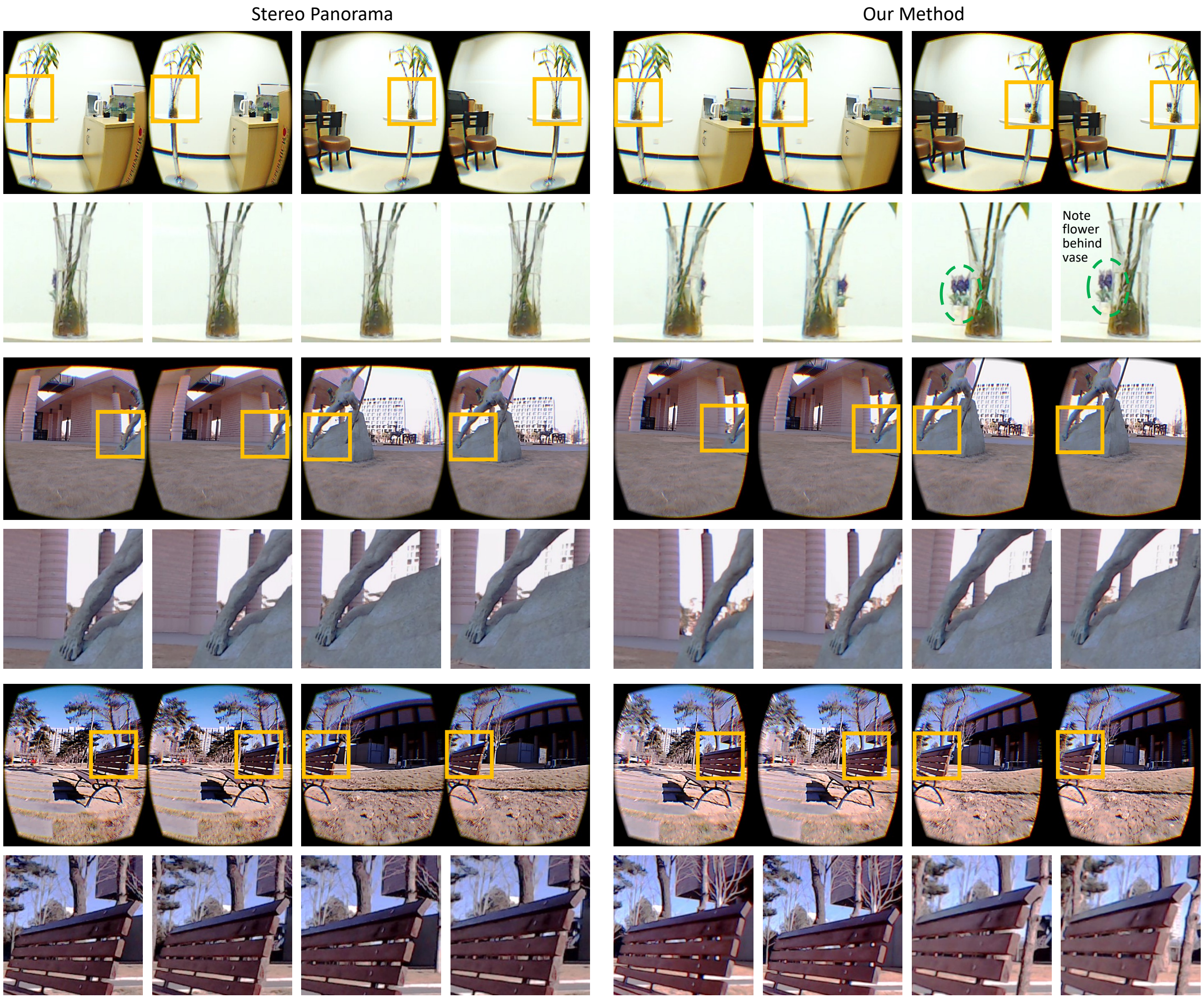

Fig. 9. Comparison of our method (right) to stereo panoramas [35] (left) on three real scenes. For each method, we show two stereo pairs viewed from different viewpoints (left/right). Zoomed crops of each view are shown below each result. **Top:** As our method supports head-motion parallax, one can see the flowers behind the vase from some viewpoints (right, circled in green), which is not the case for stereo panoaramas (see left half of figure). **Middle:** Thanks to motion parallax, one can also look behind the leg of the statue using our method. **Bottom:** When the headset is rotated, our method correctly matches the 3D rotation of the bench (with perspective effect), while the stereo panorama only applies a global 2D transform in image space. Note that the full views of these results are captured directly from the DK2 headset, which has a lower resolution than the input images, and applies chromatic aberration compensation for the headset's optics, resulting in shifted color channels and thus color fringes.

5 Results

We perform all experiments on a standard PC with a 3.6 GHz Intel Core i7 quad-core CPU, 16 GB memory, and an Nvidia GeForce 980 GPU. We use an Oculus Rift Development Kit 2 ('DK2' for short) head-mounted display, which has a display resolution of 960×1080 pixels for each eye. The valid moving range of a user is 360° horizontally and 40° vertically, and key frames are captured at an interval of 5°. Our representation requires 216 MB per scene for storing the 72×8 images and the corresponding two motion fields, which is much less than a full light-field representation. Our system achieves an average frame rate of 41.2 Hz on the DK2. In the following, we first compare our solution to existing stereo panoramas. We then evaluate the main components of our technique in terms of quality and performance. Please see our supplemental video for further results and comparisons.

5.1 Comparison

Figures 8 and 9 show the results of our approach compared to existing stereo panoramas, on synthetic and real scenes, respectively. For synthetic scenes, we render the input views using Unity, and for real scenes we use our robotic arm to capture them (see Section 4.1). In both cases, we then compute motion fields as described in Section 4.2 to complete our scene representation. We compare our results with stereo panoramas created by the state-of-the-art Megastereo technique [35]. Unlike Megastereo, our results clearly show head-motion parallax. Video comparisons are shown in the supplemental video.

Figure 8 shows a comparison on synthetic input images for two different viewing positions. When changing the viewing position, stereo panoramas only apply a homography warp, which is mostly horizontal translation of the shown imagery. In this case, there is no head-motion parallax. In our results, on the other hand, a change of viewpoint leads to head-motion parallax, which is visible in the shift of nearby and far objects relative to each other, such as the trees in the first scene. In the 'city' scene, one can clearly see the change of position of the lamp post compared to the store window behind it.

Figure 9 shows our results on three real-world scenes, compared to stereo panoramas. The 'plants' scene (top) shows a flower pot behind a vase. In our results, one can clearly see the change of position between the large vase and the small flower pot behind it. This cannot be

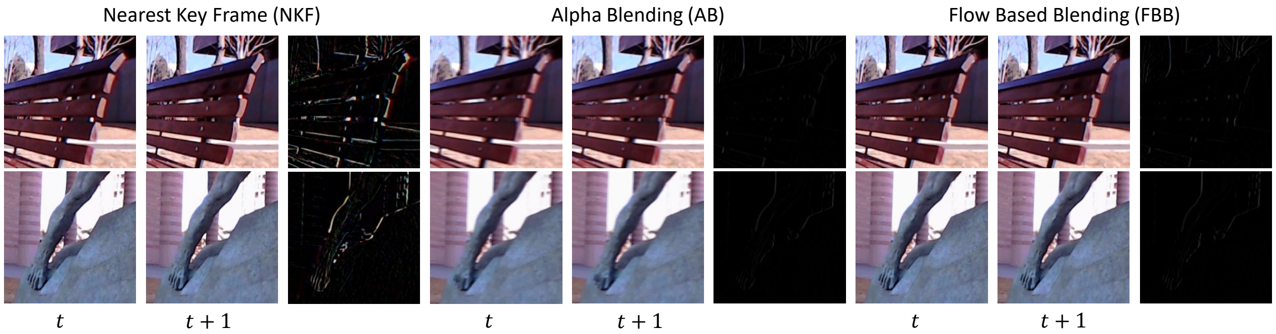

Fig. 10. Comparison of different novel-view synthesis methods. For each method, the left two columns show two consecutive synthesized images, while the right column shows their absolute difference. **Left:** Using only the nearest key frame results in clearly visible differences (known as popping artifacts) when the nearest key frame for a synthesized view changes. **Middle:** Alpha blending (between two key frames) hides popping artifacts by blending multiple synthesized images, but poor alignment results in ghosting artifacts and a blurry synthesized image. **Right:** Our flow-based blending approach (here between two key frames) results in crisp synthesized images without popping or ghosting artifacts.

observed in the stereo panorama, where the flower pot is permanently hidden behind the vase. Stereo panoramas do not provide any new viewpoints, but instead show a fixed spatial configuration for all head rotations. In the ‘statue’ scene (middle), one can see behind the statue’s leg with our approach. In the stereo panorama, the rendered images are keeping the same positional relationship. Although binocular disparity provides a sense of depth perception to viewers, head rotation is not reflected in the results beyond a simple pan, which diminishes the sense of immersion. The third scene (‘bench’) shows a bench in front of several trees. When the head-mounted display is rotated to face the bench, our approach correctly shows the 3D rotation of the bench and the resulting change in perspective, while the stereo panorama only applies a much simpler global transform. Our results reveal the shape of the bench by enlarging the part of it closest to the viewer. Stereo panoramas provide essentially the same image contents no matter where the viewer looks or moves.

Compared with stereo panoramas, which combine all information into two images, our approach takes advantage of an image-based representation that also compactly stores the depth information of a scene. Note that we are not showing any panorama results here (but we show them in the supplemental video), because they are visually similar to stereo panoramas.

5.2 Evaluation

To further evaluate our method, we compare three different solutions for synthesizing novel views:

- **Nearest key frame** (NKF) renders a novel view using only the nearest key frame I_1 ($K=1$). The target view is identical to I_1^t.
- **Alpha blending** (AB) renders a novel view using the nearest $K=2$ key frames. The target views I_1^t and I_2^t are alpha-blended without fusing the position of the corresponding pixels. This corresponds to using $p_1 = p_2 = p$ in Equation 4.
- **Flow-based blending** (FBB) renders a novel view using our full rendering method, as described in Section 4.3.

5.2.1 View Synthesis Quality

Figure 10 compares the three novel-view synthesis methods on two real scenes. To better demonstrate their differences, we consider an HMD path from the position of one key frame to another, and pick two consecutive result views t and $t+1$, where the nearest key frame switches. For a clear comparison, we only show a cropped image region. We also show video results in the supplemental video.

Using only the nearest key frame (NKF) to synthesize a novel view results in abrupt changes between frames t and $t+1$, as the nearest key frame changes. Alpha-blending (AB) between views synthesized from multiple key frames suppresses abrupt changes between t and $t+1$, because blending weights change smoothly as the viewpoint changes. However, the views synthesized from different key frames are not always perfectly aligned (e.g. at object edges), which causes ghosting artifacts that result in blurry synthesized views (see Figure 10, middle). In our approach, we perform flow-based blending (FBB), which aligns the views synthesized from different key frames before blending them. This removes the ghosting artifacts seen in the alpha-blended result, and produces a clean, crisp result, even when changing viewpoints.

Table 1. Stereo synthesis performance for different blending approaches.

View blending approach	800×600 pixels	960×1080 pixels
Nearest key frame (NKF)	44.9 Hz	20.5 Hz
Alpha blending (AB)	22.0 Hz	9.8 Hz
Naïve flow-based blending (FBB)	2.4 Hz	1.0 Hz
using pairwise motion fields (+pMF)	22.5 Hz	9.9 Hz
implemented on GPU (+GPU)	**94.5 Hz**	**41.2 Hz**

5.2.2 View Synthesis Performance

Table 1 compares the frame rates of different synthesis methods for stereoscopic rendering. Our experiments test two resolutions on the DK2: 800×600 and 960×1080 pixels (the full display resolution). Comparing these resolution levels, we see an approximately inverse linear relationship between frame rate and resolution across all methods: the frame rate is roughly halved when rendering twice as many pixels.

Alpha blending (AB) is nearly twice as expensive (with $K=2$) as using only the nearest key frame (NKF). Naïve flow-based blending, which computes the blending motion fields $f_{i\to k}^t$ using optical flow during rendering, comes at great computational cost. Using our precomputed pairwise motion fields to infer the blending motion fields $f_{i\to k}^t$ dramatically improves performance (‘FBB+pMF’ in Table 1). Our GPU implementation using Direct3D 11 HLSL (‘FBB+pMF+GPU’) achieves a further speed-up to more than 40 Hz for the full display resolution of the DK2 headset. In the supplemental video, we show that this improvement in performance does not sacrifice rendering quality compared to online flow-based blending (FBB).

5.3 Discussion

Our approach requires a great many images as input. This increases the difficulty of recording and affects the efficiency of the motion field computation. In our experiments, to capture the $72\times8\times7 = 4{,}032$ input images takes almost 2 hours, and precomputation takes almost 24 hours on a quad-core PC. The main bottleneck in our implementation is the optical flow computation [44], which takes about 99% of the time. However, note that our approach is independent of any particular flow implementation, so in principle any implementation could be used. Using a real-time optical flow method [17] could reduce our precomputation to less than one minute.

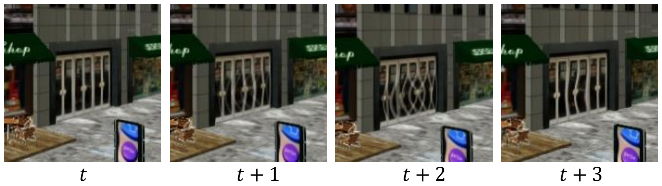

Fig. 11. View-synthesis artifacts caused by incorrect motion fields.

In addition, our scene representation has increased storage requirements compared to stereo panoramas, as it stores hundreds of key frames plus the associated patch-level disparity and pairwise motion fields. All this data is required to achieve head-motion parallax, but compression schemes used for light fields [5, 24] could likely be adapted to reduce storage requirements by an order of magnitude.

Like previous work on stereo panoramas [1, 35], the quality of our synthesized views depends mainly on the correctness of the computed optical flow. If the optical flow contains errors, the final results will likely contain artifacts. Figure 11 shows an example of such artifacts, which are caused by incorrect optical flow. In this case, a repetitive pattern leads to incorrect correspondence and hence poorly synthesized results. This problem could potentially be ameliorated by enforcing geometric consistency checks between computed flow fields.

6 Conclusion

In this paper, we presented Parallax360 – a novel 360° scene representation based on two-scale motion fields, i.e. the disparity and pairwise motion fields. Building on this representation, we propose a complete system for capturing and representing a real scene, and rendering it in a VR HMD in real time. Our approach is the first to generate novel views of a real 360° scene with head-motion parallax in real time. As shown by our results, our system generates a much more realistic 3D effect compared to stereo panoramas. Our representation is designed to handle large real scenes, where the disparity motion fields achieve implicit depth estimation by a robust curve-fitting technique that filters out noise, and the precomputed pairwise motion fields guarantee high-quality flow-based viewpoint synthesis with real-time performance. We also developed a robotic capture device to automatically and accurately capture the input images required by our approach.

Acknowledgments

This work was supported by the NSFC (No. 61671268, 61672307, 61727808), the National Key Technologies R&D Program of China (No. 2015BAF23B03) and EPSRC grant CAMERA (EP/M023281/1).

References

[1] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz. Jump: virtual reality video. *ACM Transactions on Graphics*, 35(6):198, 2016.

[2] M. Arikan, R. Preiner, and M. Wimmer. Multi-depth-map raytracing for efficient large-scene reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 22(2):1127–1137, 2016.

[3] C. Birklbauer and O. Bimber. Panorama light-field imaging. *Computer Graphics Forum*, 33(2):43–52, 2014.

[4] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

[5] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, pp. 425–432, 2001.

[6] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics*, 32(3):30, 2013.

[7] S. E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH*, pp. 279–288, 1993.

[8] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics*, 34(4):69, 2015.

[9] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi. Fusion4D: real-time performance capture of challenging scenes. *ACM Transactions on Graphics*, 35(4):114, 2016.

[10] M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. Floating textures. *Computer Graphics Forum*, 27(2):409–418, 2008.

[11] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1362–1376, 2010.

[12] M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski. Ambient point clouds for view interpolation. *ACM Transactions on Graphics*, 29(4):95, 2010.

[13] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH*, pp. 43–54, 1996.

[14] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. Casual 3D photography. *ACM Transactions on Graphics*, 36(6):234:1–15, 2017.

[15] P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. Scalable inside-out image-based rendering. *ACM Transactions on Graphics*, 35(6):231:1–11, 2016.

[16] J. Huang, Z. Chen, D. Ceylan, and H. Jin. 6-DOF VR videos with a single 360-camera. In *Proceedings of IEEE Virtual Reality (VR)*, pp. 37–44, 2017.

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, 2017.

[18] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. VolumeDeform: Real-time volumetric non-rigid reconstruction. In *ECCV*, 2016.

[19] S. B. Kang, R. Szeliski, and M. Uyttendaele. Seamless stitching using multi-perspective plane sweep. Technical Report MSR-TR-2004-48, Microsoft Research, 2004.

[20] J. Kopf, D. Lischinski, O. Deussen, D. Cohen-Or, and M. Cohen. Locally adapted projections to reduce panorama distortions. *Computer Graphics Forum*, 28(4):1083–1089, 2009.

[21] B. Krolla, M. Diebold, B. Goldlücke, and D. Stricker. Spherical light fields. In *BMVC*, 2014.

[22] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh. Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics*, 35(4):63, 2016.

[23] A. Levin and F. Durand. Linear view synthesis using a dimensionality gap light field prior. In *CVPR*, 2010.

[24] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, pp. 31–42, 1996.

[25] K. Marwah, G. Wetzstein, Y. Bando, and R. Raskar. Compressive light field photography using overcomplete dictionaries and optimized projections. *ACM Transactions on Graphics*, 32(4):46, 2013.

[26] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski. Low-cost 360 stereo photography and video capture. *ACM Transactions on Graphics*, 36(4):148, 2017.

[27] K. Mitra and A. Veeraraghavan. Light field denoising, light field superresolution and stereo camera based refocussing using a GMM light field patch prior. In *CVPR Workshops*, 2012.

[28] R. A. Newcombe, D. Fox, and S. M. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, 2015.

[29] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011.

[30] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics*, 32(6):169:1–11, 2013.

[31] S. Peleg, M. Ben-Ezra, and Y. Pritch. Omnistereo: Panoramic stereo imaging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):279–290, 2001.

[32] F. Perazzi, A. Sorkine-Hornung, H. Zimmer, P. Kaufmann, O. Wang, S. Watson, and M. Gross. Panoramic video from unstructured camera arrays. *Computer Graphics Forum*, 34(2):57–68, 2015.

[33] S. Philip, B. Summa, J. Tierny, P.-T. Bremer, and V. Pascucci. Distributed seams for gigapixel panoramas. *IEEE Transactions on Visualization and Computer Graphics*, 21(3):350–362, 2015.

[34] L. Piegl and W. Tiller. *The NURBS book*. Springer, 2012.

[35] C. Richardt, Y. Pritch, H. Zimmer, and A. Sorkine-Hornung. Megastereo: Constructing high-resolution stereo panoramas. In *CVPR*, 2013.

[36] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand. Light field reconstruction using sparsity in the continuous Fourier domain. *ACM Transactions on Graphics*, 34(1):12, 2014.

[37] H.-Y. Shum, S.-C. Chan, and S. B. Kang. *Image-Based Rendering*. Springer, 2007.

[38] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *SIGGRAPH*, pp. 299–306, 1999.

[39] H.-Y. Shum and R. Szeliski. Stereo reconstruction from multiperspective panoramas. In *ICCV*, 1999.

[40] H.-Y. Shum and R. Szeliski. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *International Journal of Computer Vision*, 36(2):101–130, 2000.

[41] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[42] S. Wanner and B. Goldluecke. Variational light field analysis for disparity estimation and super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(3):606–619, 2014.

[43] K. Xu, H. Huang, Y. Shi, H. Li, P. Long, J. Caichen, W. Sun, and B. Chen. Autoscanning for coupled scene reconstruction and proactive object analysis. *ACM Transactions on Graphics*, 34(6):177, 2015.

[44] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2012.

[45] L. Zelnik-Manor, G. Peters, and P. Perona. Squaring the circle in panoramas. In *ICCV*, 2005.

[46] Y. Zhang, W. Xu, Y. Tong, and K. Zhou. Online structure analysis for real-time indoor scene reconstruction. *ACM Transactions on Graphics*, 34(5):159, 2015.

[47] Z. Zhang, Y. Liu, and Q. Dai. Light field from micro-baseline image pair. In *CVPR*, 2015.