Image-Based Scene Representations for Head-Motion Parallax in 360° Panoramas

Tobias Bertel[1][0000−0002−6279−2969], Feng Xu[2][0000−0002−0953−1057], and Christian Richardt[1][0000−0001−6716−9845]

[1] University of Bath
T.B.Bertel@bath.ac.uk, christian@richardt.name
https://richardt.name/
[2] Tsinghua University
feng-xu@tsinghua.edu.cn

Abstract. Creation and delivery of "RealVR" experiences essentially consists of the following four main steps: capture, processing, representation and rendering. In this chapter, we present, compare and discuss two recent end-to-end approaches, Parallax360 by Luo *et al.* [9] and MegaParallax by Bertel *et al.* [3]. Both propose complete pipelines for RealVR content generation and novel-view synthesis with head-motion parallax for 360° environments.

Parallax360 uses a robotic arm for capturing thousands of input views on the surface of a sphere. Based on precomputed disparity motion fields and pairwise optical flow, novel viewpoints are synthesized on the fly using flow-based blending of the nearest two to three input views which provides compelling head-motion parallax.

MegaParallax proposes a pipeline for RealVR content generation and rendering that emphasizes casual, hand-held capturing. The approach introduces view-dependent flow-based blending to enable novel-view synthesis with head-motion parallax within a viewing area determined by the field of view of the input cameras and the capturing radius.

We describe both methods and discuss their similarities and differences in corresponding steps in the RealVR pipeline and show selected results. The chapter ends by discussing advantages and disadvantages as well as outlining the most important limitations and future work.

Keywords: 360° scene capture · scene representation · head-motion parallax · image-based rendering · novel-view synthesis

1 Introduction

A very important aspect to provide *immersive* VR experiences is head-motion parallax. *Motion parallax* occurs whenever we move our head, e.g. when looking around, driving a car or riding a train. When riding a train and looking out of the window, you notice that closer objects will pass much faster than objects farther away, such as a mountain in the distance. This effect is called motion parallax and is an essential monocular depth cue for the human visual system.

This chapter presents two recent methods that build upon an image-based scene representation containing dense imagery and which are able to provide head-motion parallax in real-world environments by employing a flow-based view synthesis.

On one hand, Parallax360 [9] supports full head-motion parallax and relies on a non-casual capturing and time-consuming processing stage. On the other hand, MegaParallax [3] provides restricted head-motion parallax and builds on a casual capturing stage and an additional step for extrinsic calibration via Structure-from-Motion (SfM) [17] and frame registration. Both methods use optical flow to establish dense (per-pixel) correspondences between *reference viewpoints* or *key frames*. Lastly, both approaches employ different *flow-based* blending procedures which allow to render novel viewpoints in high-quality and in real-time.

Creating content for real-world VR applications is a very interesting research field. One day, VR content will be as ordinary as an image or an video is today.

2 Related Work

Image-based rendering (IBR) or novel-view synthesis at its core can be formulated as follows: *Given a set of reference views of a scene, how can novel views be inferred from them?* Since it is not practical to sample the space of possible viewpoints (plenoptic function [1]) very densely, e.g. using light fields [8], novel or missing viewpoints have to be predicted or interpolated sufficiently fast at runtime to change virtual viewpoints smoothly. IBR methods have been commonly categorized according to the type of geometry used by the representation to perform novel-view synthesis (see Section later in this chapter).

The state of the art for capturing and displaying 360° real-world environments in terms of visual quality and correctness is based on *explicit* scene reconstruction [7,6]. However, 3D reconstruction of arbitrary environments with a single moving camera is extremely challenging. To give an example, dynamic objects like cars, people, animals, plants etc. may move during the capture. Furthermore, shiny or specular scene objects, such as cars, metals, polished surfaces, glass, water, etc. have a view-dependent appearance which leads to incorrect feature matches and thus to erroneous 3D reconstructions.

The commercial standard for real-world VR content generation, transmission and playback [2,16] is based on omnidirectional stereo [14,15], which is *not* relying on any explicit geometry reconstruction, but image stitching techniques [21] tweaked with *implicit* geometry, such as optical flow. The most promising work with respect to VR video was recently published by Facebook [13] and is based on IBR with explicit geometry.

The subject of 3D reconstruction is not discussed in this chapter. The reader finds excellent information about that in Peter Hedman's chapter in this very book. The presented approaches explicitly avoid 3D reconstruction by relying on densely sampled imagery. The set of all reference viewpoints and their corresponding pixels can be seen as a large database of light rays, assuming an infinitely small aperture as its the case in pinhole images. Only the most relevant work to understand and motivate Parallax360 and MegaParallax is addressed in the remainder of this section.

2.1 The Plenoptic Function

The plenoptic function [1] describes the incident radiance at a 3D point (x, y, z) from a certain direction (θ, ϕ), in 2D spherical coordinates, with wavelength λ at time t. Thus overall, the function has 7 dimensions. The wavelength is usually discretized in a 3-channel RGB texture, and it is assumed in this chapter that time is fixed. The plenoptic function thus becomes a 5D function depending on space (3D) and direction (2D): $\mathbf{L}(x, y, z, \theta, \phi)$.

Every pinhole image taken of the real-world represents a finite set of samples of the plenoptic function assuming the optical center of the camera as (x, y, z) and the field of view as a range over θ and ϕ. The fundamental goal of all image-based rendering approaches is to reconstruct the plenoptic function $\mathbf{L}$ *continuously* given only finitely many samples of it.

Plenoptic Modelling [10] is an IBR framework describing (1) sampling, (2) reconstruction, and (3) resampling of the plenoptic function to create novel viewpoints from a set of monoscopic cylindrical panoramas. Novel viewpoints show correct perspectives and visibility without reconstructing any scene geometry explicitly, but only estimating optical flow between the reference images. The downsides of the method are the non-casual capture of cylindrical images and the required density of viewpoints to keep optical flow estimation reliable.

Light Field Rendering [8] presents a 4D representation of the 5D plenoptic function called a light field by parameterizing plenoptic samples using rays connecting two planes that can be described by two 2D coordinates from the respective planes. Their input images are sampled so densely that reconstructing a desired viewpoint is performed by solely looking up and bilinearly (or quadrilinearly, if a focal plane is used additionally to the camera plane) blending reference pixels. The main downside of the method is the large memory footprint of light fields. Dozens of images captured with a gantry are necessary to obtain a viewing space of a few centimeters, in which head-motion is supported.

Rendering with Concentric Mosaics [19] present a 3D representation of the 5D plenoptic function which is limited to translation in a plane. Multiple concentric circles are sampled using a *slit-image* approach as often used in the image stitching community [14,21,15]. Every slit image is naturally described by radius, rotation angle and vertical elevation. A desired viewpoint is synthesized by mosaicking reference slit images. This work was the first to report the effect of *vertical distortion*, which occurs if a desired viewpoint is stitched with slit images that were captured far away from the desired optical center. The suggestion to use a constant-depth proxy to address vertical distortion can be seen as constant composition surface in image stitching algorithms [21]. Vertical distortion and the non-casual, time-consuming capturing process are the main downsides of this method.

2.2 Image-Based Rendering (IBR)

approaches all aim for reconstructing the plenoptic function continuously given a finite set of reference viewpoints and estimated *correspondences* among these viewpoints.

IBR methods can be categorized according to the type of geometry which is contained in the representation and used in the view synthesis stage [18]. We will highlight relevant aspects of IBR in this chapter which give important motivation and context for the presented papers.

No Geometry Light field rendering [8] and concentric mosaics [19] do not rely on any reconstructed geometry to synthesize novel viewpoints.

Implicit Geometry Pixel correspondences between a pair of images implicitly model geometric relations among the depicted scene objects. Motion vectors of close objects will have a larger magnitude than objects farther away due to motion parallax. Plenoptic modelling [10] uses optical flow between reference views to establish dense pixel correspondences needed for rendering. Megastereo [15] utilizes a casual capturing setup and performs flow-based blending in order to mosaic omnidirectional stereo panoramas [14] assuming a constant-depth composition surface.

The papers we focus on in this chapter are both IBR methods relying on *implicit* geometry.

Explicit Geometry As soon as an IBR approach relies on geometry, e.g. in the form of depth maps or a global scene proxy or mesh, it is called *explicit*. The main idea of these methods is that desired viewpoints can be estimated by *reprojecting* reference views into the desired view using the scene geometry.

Unstructured Lumigraph Rendering (ULR) [4] postulates desirable properties any IBR system should have. If given a sufficient amount of images, it turns into lumigraph rendering [5]. Otherwise, if given an unstructured set of input images with a high-quality geometric proxy, it turns into view-dependent texture mapping.

Overbeck *et al*. present a end-to-end VR pipeline for full head-motion parallax [12]. The capture is non-casual and rendering involves explicit geometry in the form of depth maps. The results are visually outstanding, but the representation is large.

It was shown that image edges must align well with explicitly reconstructed geometry in order to minimize visual artefacts along depth edges when performing novel-view synthesis [7]. Hedman *et al*. further shows how to reconstruct a set of casually captured fish-eye images into global scene proxies and statically texture them by proposing a state-of-art plane-sweep algorithm and efficient cost-volume filtering [6].

2.3 Learning-Based Approaches

Deep neural networks define the state of the art for many computer vision tasks, e.g. optical flow estimation, segmentation, classification, depth prediction from single images, and many more. The essence of all learning-based approaches is *data*. Many of the most successful networks are trained in a *supervised* manner, meaning that ground-truth labels for the training data are required. For example, to train a supervised cat detector, a network might see a set of images containing cats and images containing no cats. For each incorrect prediction in an epoch (iterating once through the entire training dataset), a loss is accumulated which informs the neural network about its mistakes, i.e. wrong

predictions. *Unsupervised* learning approaches train on unlabeled data and try to infer more general structures inherent in the observed data.

The most promising learning-based approaches used for IBR rely on physically motivated *models*[1] which can be trained to obtain powerful scene representations such as multiplane images (MPIs) as introduced by Zhou *et al.*'s stereo magnification work [22]. The authors train a network from data which is mined from YouTube videos and fully automatic scene reconstructions using SfM, with the goal of extrapolating novel viewpoints beyond the baseline of a stereo image captured with a dual camera of a smartphone. The core idea of the MPI representation is that a scene volume can be modeled by a set of fronto-parallel semi-transparent layers[2] which can be rendered from novel viewpoints.

The current state of the art of learned image-based representations suitable for head-motion parallax is local light field fusion [11]. The method generates *local* light fields by taking a set of casually captured images as input and predicting MPIs for each image with its four closest neighbors. At runtime, neighboring light fields are *fused* to continuously render novel viewpoints. The biggest limitations of the representation is its size and its current restriction to indoor environments.

3 Methods

This section gives an overview of the presented methods, which both rely on a dense sampling of key frames or reference viewpoints captured with a single moving camera. In both works, novel views are synthesized by combining viewpoints using some sort of flow-based blending *without* relying on any type of explicit geometry like depth maps or meshes. Both methods assume that all input views share the same intrinsic calibration and both register the central rays of captured viewpoints to *radial* directions of a camera manifold, i.e. a circle or a sphere, which is technically done by parameterizing the optical center and central ray using polar or spherical coordinates respectively.

3.1 Parallax360

The scene representation of Parallax360 consists of uniformly sampled key frames on a sphere (see Figure 1 (a)), captured using a robot arm (hence no extrinsic calibration is needed), disparity motion field for each key frame, calculated with the relative frames around the key frames, as well as dense bidirectional optical flow between key frames. A novel viewpoint is synthesized by determining the K nearest key frames (in practice $K = 2, 3$), followed by a patch-wise flow-based warping operation concluded by an alpha-blending step.

[1] E.g. a model aiming to minimize the reprojection error of a set of images as commonly used in bundle-adjustment.

[2] This concept was first used by Walt Disney in the early stages of cartoon productions when artists moved semi-transparent layers relatively to each other to create new scenes.

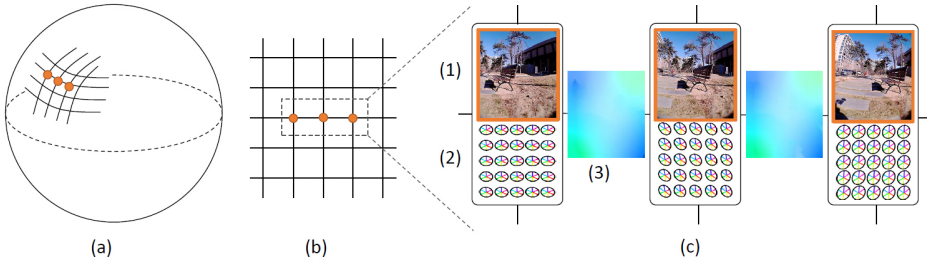

Fig. 1: Parallax360 representation: (a) Sampling sphere with three key frames (in orange), (b) key frames on longitude/latitude grid, (c) three main components: (1) key frames, (2) curve-based disparity motion fields, and (3) optical flow between key frames.

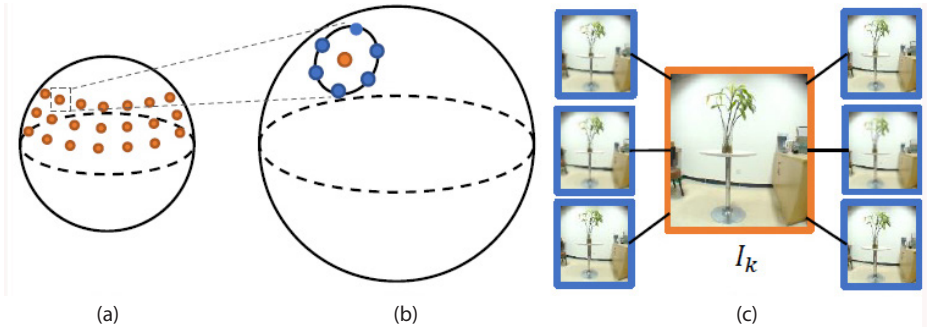

Fig. 2: Parallax360 capture: (a) key frames, (b) disparity motion fields to 6-node ring neighborhood, (c) zoom in to (b).

Capture A *robot arm* captures thousands of plenoptic samples (pinhole images) and support images in a 6-node ring neighborhood (relative frames), as shown in Figure 2, along the radial directions of a spherical surface. Since the robot's positioning is very accurate, there is no need for external calibration, i.e. reconstruction of camera geometry, after the capturing procedure. The capturing process for Parallax360 datasets is non-casual and time-consuming, e.g. capturing a dataset (key frames and relative frames) is done in almost 2 hours.

Processing The core of the Parallax360 processing is the computation of *curve-based* disparity motion fields and pairwise optical flow used for novel-view synthesis. Each disparity motion field $\mathbf{f}_1, \ldots, \mathbf{f}_6$ is computed from a set of 7 frames: 1 key frame and its 6 relative frames (see Figure 2 (b,c)). Flow vectors $\mathbf{f}_i(\mathbf{p})$ per pixel $\mathbf{p}$ are aggregated into image patches $\mathbf{P}$ of size 8×8 pixels by averaging them. The 6 initial disparity motion fields of a key frame are converted into a curve-based motion representation, modeled by a fitted ellipse and 6 polar angles (see Figure 3 (e)). For every patch $\mathbf{P}$ of a key

frame, an ellipse $\mathbf{E}_P$[3] is computed by least-squares fitting. Five 2D points are needed for describing an ellipse as a quadric surface, and six motion vectors to relative frames are available. The endpoints of the motion vectors originating from the center of a patch are used to fit the ellipse. The motion field generation takes up to 24 hours for a single dataset on a quad-core computer.

Representation The fundamental component of Parallax360 datasets is the *curve-based* motion representation (see Figure 3). The parameters of the fitted ellipse $\mathbf{E_P}$ and polar angles $\theta^i_{\mathbf{P}}$ are used to encode the magnitudes and directions of the motion vectors respectively. Every patch $\mathbf{P}$ of each key frame is associated with a fitted ellipse and 6 polar angles. The authors state that the estimation errors of the individual motion vectors can be alleviated by fitting the ellipse to the endpoints of the motion vectors. Key and relative frames are parameterized in spherical coordinates.

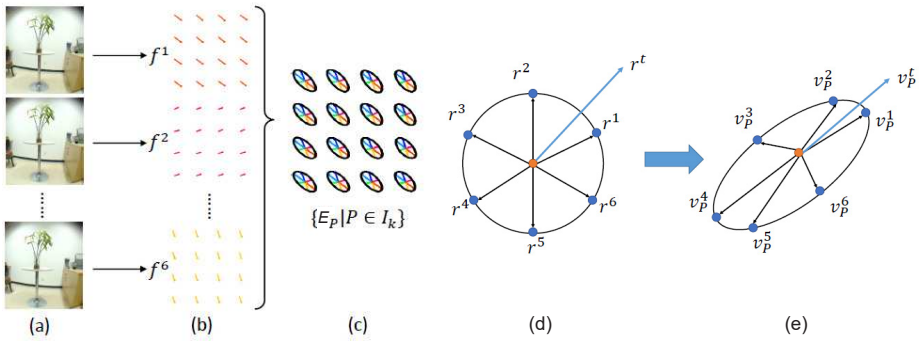

Fig. 3: Parallax360 disparity motion fields (a,b) and curve-based motion fields (c,d,e): (a) Relative frames, (b) corresponding motion fields, (c) disparity motion curves, (d) key and relative frames $\mathbf{r}^i$, (e) motion vectors $\mathbf{v}^i_{\mathbf{P}}$ for a patch $\mathbf{P}$.

Rendering A target viewpoint $\mathbf{r}^t$ in Parallax360 is assumed to be tangential[4] to the spherical imaging surface exactly like the key frames. The authors consider the two or three closest key frames to synthesize a novel viewpoint (see Figure 4).

For each key frame, the direction of the target viewpoint $\mathbf{r}^t$ can be described by the two closest motion vectors expressed by the ellipse $\mathbf{E_P}$ and the polar coordinates θ_P of the nearest two relative frames, e.g. $\mathbf{r}^1$ and $\mathbf{r}^2$ in Figure 3 (d). $\mathbf{r}^t$ can thus be written as a convex combination of polar coordinates:

$$\theta(r^t) = \alpha\theta^1_{\mathbf{P}} + \beta\theta^2_{\mathbf{P}} \tag{1}$$

[3] The ellipse is denoted $\mathbf{C}_P$ in the original paper, but this conflicts with the camera centers $\mathbf{C}_i$ used in MegaParallax.

[4] Image planes are tangential to the sphere meaning orthogonal to the surface normal.

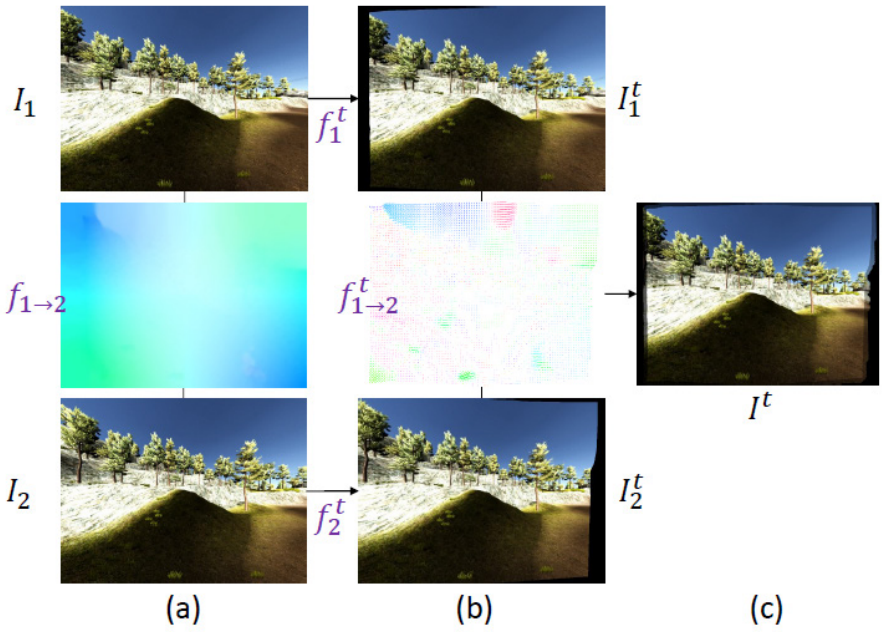

Fig. 4: Parallax 360 view synthesis: (a) Two closest key frames $\mathbf{I}_1, \mathbf{I}_2$ and optical flow $\mathbf{f}_{1\to 2}$, (b) key frames warped $\mathbf{I}_1^t, \mathbf{I}_2^t$ into the target viewpoint $\mathbf{r}^t$ and blending motion field $\mathbf{f}_{1\to 2}^t$, (c) image of target viewpoint $\mathbf{I}^t$.

$\alpha + \beta = 1$ and $\theta(\mathbf{r}^t)$ denotes the polar angle of the 2D vector $\mathbf{r}$. For a patch $\mathbf{P}$ in the key frame, the motion vector is computed by using

$$\mathbf{v}_{\mathbf{P}}^t = \frac{\|\mathbf{r}^t\|}{\|\mathbf{r}^1\|} \cdot \mathbf{E}_{\mathbf{P}}(\alpha\theta_{\mathbf{P}}^1 + \beta\theta_{\mathbf{P}}^2) \tag{2}$$

Here, $\mathbf{E}_{\mathbf{P}}(\theta)$ is a motion vector from the center of the ellipse to the point on the ellipse with polar angle θ. The parameters α and β are further used to synthesize *intermediate* target images $\mathbf{I}_k^t, k \in \{1, 2, 3\}$ by warping patches of the corresponding key frames:

$$\mathbf{I}_k^t(\mathbf{p}) = \mathbf{I}_k\left((\mathbf{f}_k^t)^{-1}(\mathbf{p})\right), \tag{3}$$

where $(\mathbf{f}_k^t)^{-1}$ maps the pixels of the target frame $\mathbf{I}^t$ into the k-th key frame $\mathbf{I}_k$. Note that the representation is based on image patches $\mathbf{P}$ but the target frame $\mathbf{I}^t$ is synthesized per pixel. The aggregated or patched motion vectors are smoothly propagated per pixel by using bilinear interpolation.

To avoid computing the flow field online for blending the intermediate images, Parallax360 uses the following strategy:

1. Map pixels from the target view $\mathbf{I}^t$ into the key frame $\mathbf{I}_i$ by applying the inverse of the disparity motion field $\mathbf{f}_i^t$,

2. apply the pairwise motion field $\mathbf{f}_{i\to k}$ to key frame $\mathbf{I}_k$
3. use $\mathbf{f}_k^t$ to end up in the coordinates of the target image $\mathbf{I}_k^t$:

$$\mathbf{f}_{i\to k}^t = \mathbf{f}_k^t \oplus \mathbf{f}_{i\to k} \oplus (\mathbf{f}_i^t)^{-1}, \tag{4}$$

where $\oplus$ denotes the operator to concatenate mappings. With this strategy, we leverage the pre-computed optical flow $\mathbf{f}_{1\to 2}$ to achieve real-time rendering of $\mathbf{I}^t$.

Note that it is useful to think about the rendering as a *backward warping* process, in which the target image plane is the novel viewpoint $\mathbf{r}^t$ and the source plane is one of the nearest key frames $\mathbf{I}_k$. Since the novel viewpoint can be expressed in each patch of each key frame by using the fitted ellipse $\mathbf{E}_k$ and a polar angle, intermediate images can be created according to Equation 3. In order to blend smoothly, optical flow is approximated for the intermediate images $\mathbf{I}_k^t$ as shown in Equation 4 and used for *flow-based* blending. The authors report *ghosting* artifacts (see Figure 7 right) when not using the flow field $\mathbf{f}_{1\to 2}^t$. The issue becomes more severe for challenging viewpoints that contain scene objects close to the camera because of their different perspectives in the key frames.

3.2 MegaParallax

The scene representation of MegaParallax consists of casually captured key frames on a circle (see Figure 5, Capture), e.g. hand-held using a consumer camera or mobile device, extrinsic calibration obtained by SfM and bidirectional optical flow between the key frames.

A novel viewpoint is synthesized by reconstructing individual camera rays of a desired viewpoint. First, for each ray, the enclosing pair of key frames is determined. Second, the ray reconstruction itself uses a flow-based blending of a pair of pixels, stemming from the camera pair (one pixel per camera), which correspond according to a simple proxy geometry, e.g. a plane parallel to the desired camera's image plane, but much farther away (see Figure 8).

Capture MegaParallax datasets share exactly the same input requirement as suggested in Megastereo [15]. *Hand-held* cameras can be used to acquire datasets for 360° stereo panoramas. The suitability of the captured video to get successfully processed into a Megastereo or MegaParallax dataset is fundamentally determined by the quality of the extrinsic calibration. The capturing procedure itself takes only about 10 seconds.

There is active research that focuses on making the reconstruction of the desired egocentric camera paths more robust [20]. The dominance of rotation compared to translation makes egocentric camera paths very hard to reconstruct.

Processing As shown in Figure 5, the processing involves the following five steps:

1. Estimate camera extrinsics using SfM, e.g. COLMAP [17],
2. fit a circle to the optical centers,

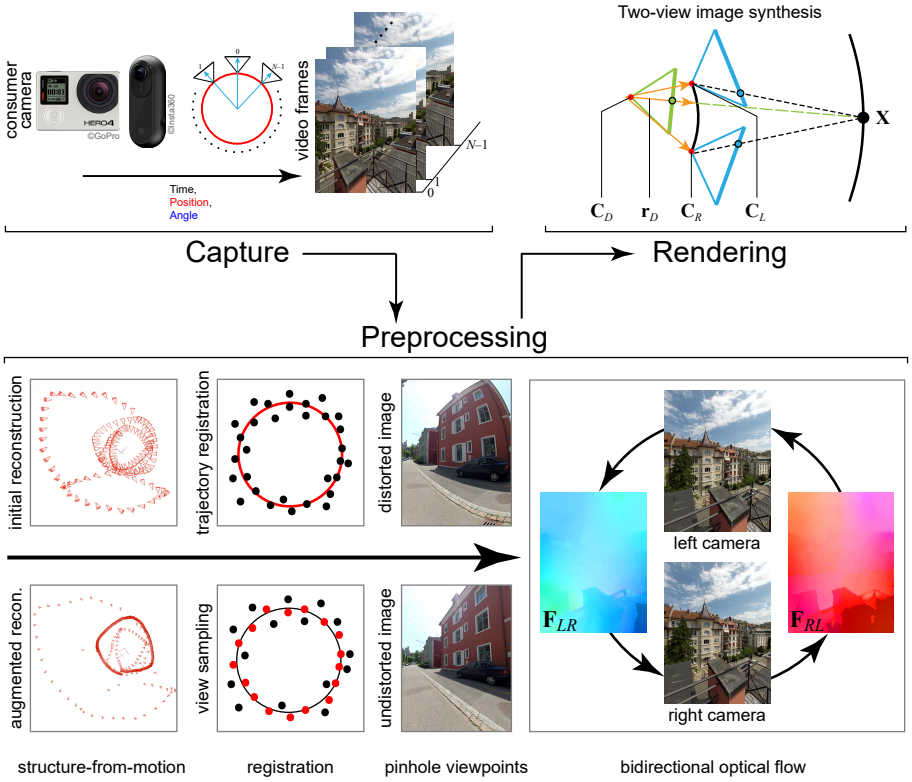

Fig. 5: MegaParallax pipeline: Capture: Video recorded along circular trajectory. Preprocessing: Estimate extrinsics, fit ideal trajectory, allows simple key frame registration via polar coordinates, undistort key frames, bidirectional flow for each pair of neighboring cameras. Rendering: Viewing rays are reconstructed using view-dependent flow-based blending.

3. register viewpoints (a.k.a. input, reference or key frames) to the circle identifying the optical axis of each viewpoint with a polar angle ϕ,
4. undistort images to obtain pinhole images, and lastly
5. compute bidirectional optical flow for each reference frame towards its left and right neighbors.

The bottleneck of this stage is the estimation of the camera extrinsics (1) since the specific egocentric, inside-out camera path makes the reconstruction very challenging. Bertel *et al.* perform the extrinsic reconstruction in two passes. The first pass is performed on a subset of frames to increase the baselines between neighboring pairs of cameras. The second pass registers the remaining frames to the reconstruction from the first step. The viewpoints are picked by uniformly sampling polar angles induced by the circle. Estimating the extrinsics, i.e. performing a sparse SfM reconstruction, for a

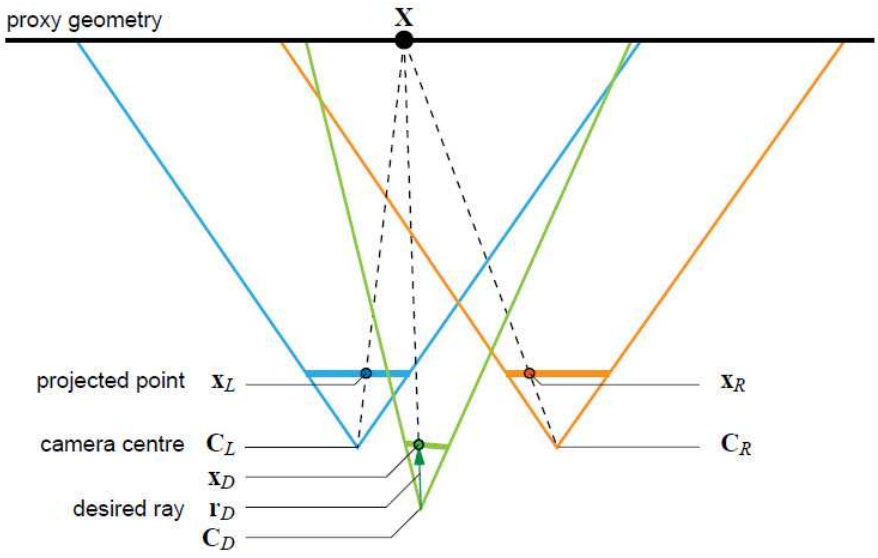

Fig. 6: MegaParallax view synthesis: The camera ray $\mathbf{x}_D$ is reconstructed by projecting a world point $\mathbf{X}$ into the surrounding pair of cameras $\mathbf{C}_L$ and $\mathbf{C}_R$.

dataset takes between 30 and 60 minutes. The used two-pass strategy[5] allows for quick reconstructions. A few iterations of bundle adjustment can be applied to refine the final reconstruction. Finally, computing optical flow (5) is reliable as long imaged scenes are mostly diffuse and static, and source and target images are sufficiently close, i.e. having a small baseline and share similar orientations.

Representation The representation of a MegaParallax dataset is very simple. The following data must be provided for each viewpoint at runtime: (1) the corresponding reference frame $\mathbf{I}_k$ and flow fields to left and right neighbor frames ($\mathbf{F}_{k\to k-1}$ and $\mathbf{F}_{k\to k+1}$ respectively), and (2) the projection matrix. The fitted circle and a linear array indexed by the polar coordinates of the viewpoints are needed to enable fast lookup operations at runtime.

A static cylindrical or dynamic planar scene geometry is taken as proxy geometry for the rasterization pipeline of OpenGL. The cylinder mimics previous panorama stitching algorithms [14,15,21] whereas a fronto-parallel plane in front of a desired viewpoint has been used in light field rendering [8].

Rendering The rendering procedure used in MegaParallax is depicted in Figure 6. A pixel $\mathbf{x}_D$ in the desired view is a convex combination of pixels $\mathbf{x}_L$ (left viewpoint) and $\mathbf{x}_R$ (right viewpoint):

$$\mathbf{I}_D(\mathbf{x}_D) = (1-\alpha)\cdot\mathbf{I}_L(\mathbf{x}_L) + \alpha\cdot\mathbf{I}_R(\mathbf{x}_R), \tag{5}$$

[5] Originally proposed in Marc Pollefeys' PhD thesis.

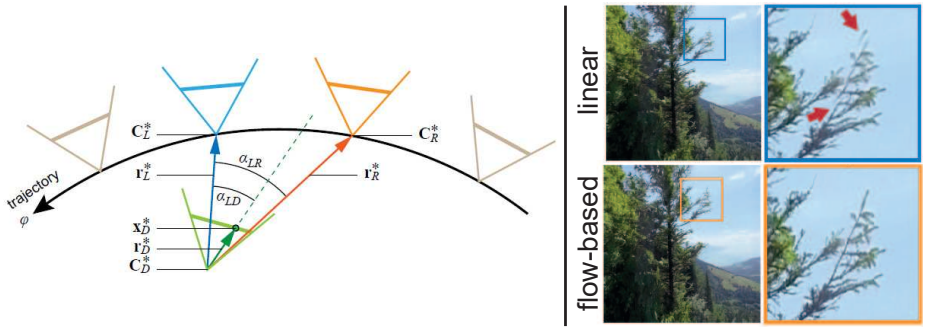

Fig. 7: MegaParallax view-dependent blending weight and ghosting: **Left:** View-dependent blending weight determined by desired camera ray and enclosing camera pair. **Right:** Flow-based blending is used to avoid ghosting artefacts.

The blending weight α depends on the relative angle α_{LD} between the vectors $\mathbf{r}_L^*$ and $\mathbf{r}_R^*$ connecting the optical centers of the left $\mathbf{C}_L$ and right $\mathbf{C}_R$ viewpoints with the optical center of the desired viewpoint and the desired camera ray (see Figure 7, left). Note that the blending weights – and hence overall color – are computed independently per pixel of the desired viewpoint, which allows for extrapolation of novel viewpoints.

The use of an inaccurate proxy geometry may produce large reprojection errors, which reveal themselves as blurry artefacts or texture misalignments, such as ghosting (see Figure 7, right). Megastereo [15] proposes flow-based ray interpolation to overcome these artefacts.

MegaParallax proposes *view-dependent* flow-based blending, which is not solely restricted to a fixed viewing circle as applied in Megastereo's casual ODS approach. To alleviate ghosting artefacts, MegaParallax uses *flow-corrected* pixel coordinates $\mathbf{x}_L^*$ and $\mathbf{x}_R^*$ to sample source pixels from the left $\mathbf{I}_L$ and right $\mathbf{I}_R$ reference images, respectively, to synthesize a pixel $\mathbf{x}_D$ in the desired image $\mathbf{I}_D$:

$$\mathbf{I}_D(\mathbf{x}_D) = (1-\alpha)\cdot\mathbf{I}_L(\mathbf{x}_L^*) + \alpha\cdot\mathbf{I}_R(\mathbf{x}_R^*). \tag{6}$$

The flow-corrected pixel coordinates are obtained as follows:

1. A plane-induced displacement between the projections $\mathbf{x}_L$ and $\mathbf{x}_R$ is performed:

$$\mathbf{v}_{LR} = \mathbf{x}_R - \mathbf{x}_L \quad \text{and} \quad \mathbf{v}_{RL} = \mathbf{x}_L - \mathbf{x}_R. \tag{7}$$

2. The motion-corrected flow vectors are obtained using:

$$\mathbf{F}_{LR}^*(\mathbf{x}_L) = \mathbf{v}_{LR} - \mathbf{F}_{LR}(\mathbf{x}_L) \quad \text{and} \tag{8}$$
$$\mathbf{F}_{RL}^*(\mathbf{x}_R) = \mathbf{v}_{RL} - \mathbf{F}_{RL}(\mathbf{x}_R). \tag{9}$$

3. Scaling these displacements yields the flow-corrected image coordinates:

$$\mathbf{x}_L^* = \mathbf{x}_L + \alpha\cdot\mathbf{F}_{LR}^*(\mathbf{x}_L) \quad \text{and} \tag{10}$$
$$\mathbf{x}_R^* = \mathbf{x}_R + (1-\alpha)\cdot\mathbf{F}_{RL}^*(\mathbf{x}_R). \tag{11}$$

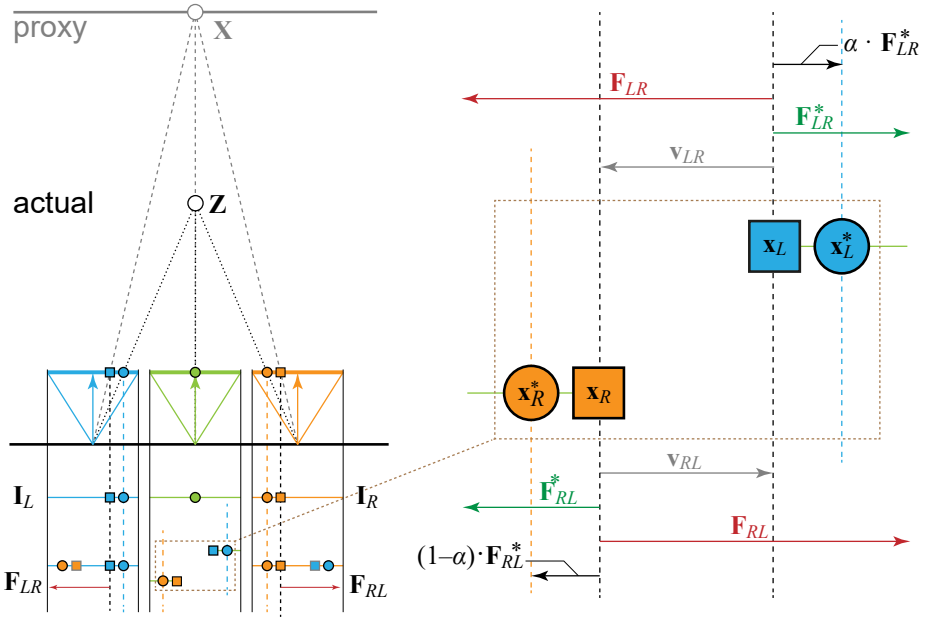

Fig. 8: MegaParallax's view-dependent flow-based blending enables motion parallax without ghosting artefacts. More description in the text.

Note that view-dependent flow-based blending resolves ghosting in the desired viewpoints, but does not necessarily produce correct perspectives in all cases due to vertical distortion. The scaled displacements $\alpha \cdot \mathbf{F}^*_{LR}$ and $(1-\alpha) \cdot \mathbf{F}^*_{RL}$ only push the initial projections $\mathbf{x}_L$ and $\mathbf{x}_R$ in the *correct* directions, as illustrated in Figure 8. The method works well if the angular baseline[6] between images is sufficiently small with respect to the closest scene object, and optical flow is sufficiently smooth.

The rendering speed is very fast ($>$ 200 fps) since all flow-fields are precomputed and thus only light-weight computations, mostly texture lookups, need to be performed at runtime. The rendering strategy delivers high-quality results and supports desired views with wide field of views by design. The per-pixel blending allows for view extrapolation, which occurs whenever a viewpoint is synthesized *within* the camera circle. This can be seen at best in the supplemental video of MegaParallax[7] when translational camera motion is compared against Parallax360.

4 Results

This section shows some selected results shown in the presented papers. Luo *et al.* show comparisons between omnidirectional stereo (ODS) panoramas [14,15] and Par-

[6] Angle between neighboring viewpoints, e.g. 180 viewpoints sampled uniformly on a circle yields an angular baseline of 2°.

[7] https://richardt.name/megaparallax.mp4

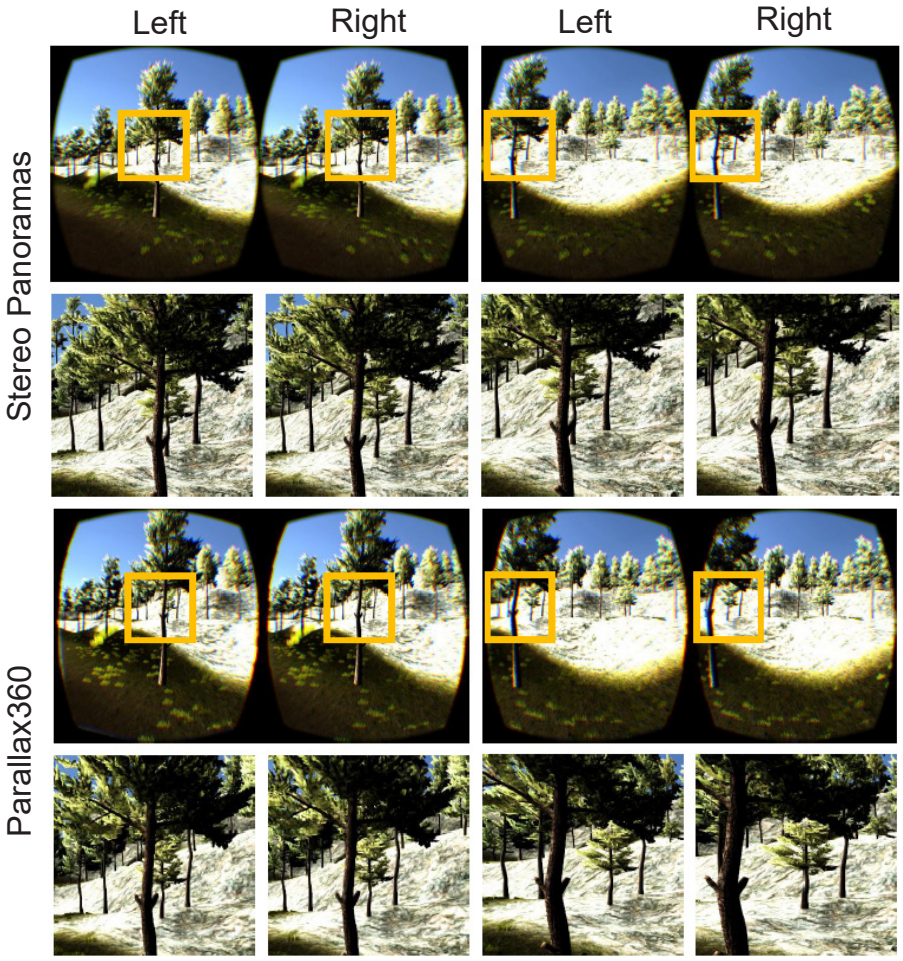

Fig. 9: Parallax360 results: A comparison between stereo panoramas and Parallax360 results. More description in the text.

allax360 in Figure 9. The first row depicts two viewpoints (left and right view) with different head orientations obtained from a stereo panorama. The second row shows close-ups of the rectangles depicted in the first row. Note that there is no relative motion between scene objects. The third row shows two stereo viewpoints obtained by Parallax360. The fourth row shows close-ups in spirit of the second row. ODS panoramas provide binocular disparity for head rotation but do not support head translation and thus neither support motion parallax by design. Note the relative displacements of the trees observed in the fourth row of Figure 9, caused by motion parallax.

Bertel *et al*. show comparisons between Unstructured Lumigraph Rendering (ULR) [4], Megastereo [15] and MegaParallax in Figure 10. First row: ULR provides motion

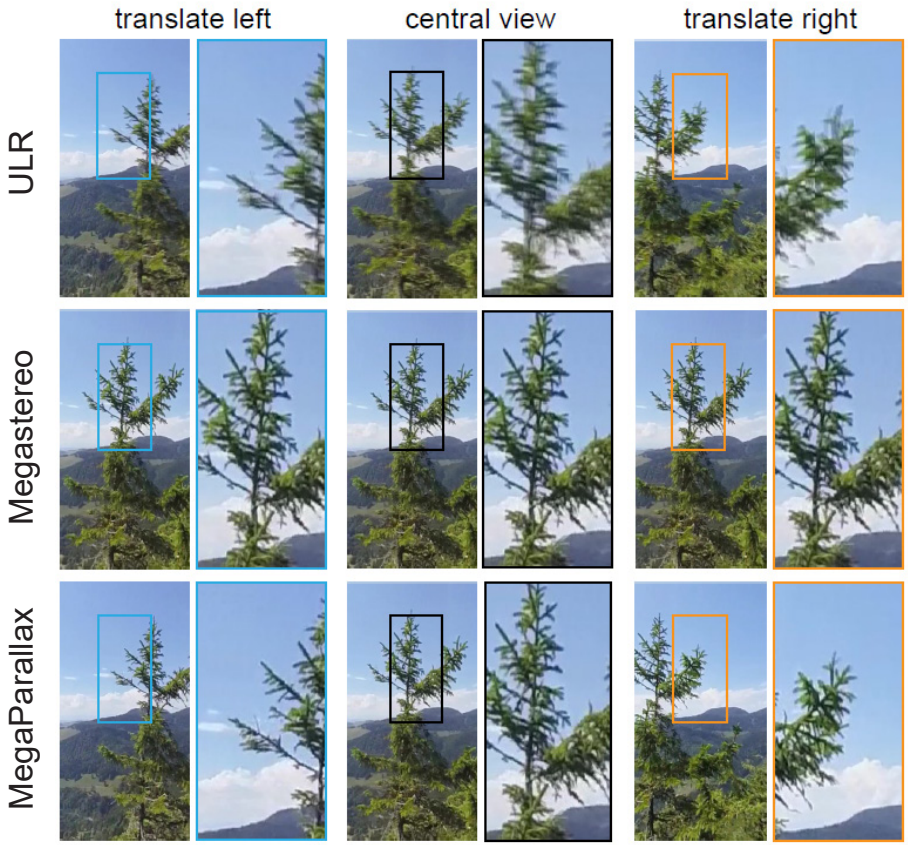

Fig. 10: MegaParallax results: Comparison between ULR (motion parallax but blurry), Megastereo (no motion parallax but crisp) and MegaParallax (motion parallax and crisp). More description in the text.

parallax, but leads to blurry rendering artefacts (ghosting) which is expected when using an inaccurate scene proxy. Second row: Megastereo shows no motion parallax, but the viewpoints show visually crisp results (no ghosting). Third row: MegaParallax combines the best of both approaches, namely motion parallax without introducing ghosting artefacts.

5 Discussion

We now discuss the most important aspects of the two methods described in this chapter and focus on the need of compelling visual content that is suitable for RealVR experiences, i.e. content that provides head-motion parallax at runtime.

5.1 Capturing

Parallax360 uses a robot arm to capture thousands of input images. The capturing procedure takes less than two hours. MegaParallax, on the other hand, needs only hundreds of input images and the capturing takes about 10 seconds. The input images are extracted from a continuous sweep of a hand-held camera (see Figure 5 Capture). However, note that Parallax360 densely captures a spherical imaging surface (see Figure 2), while MegaParallax just captures one single circle.

The fundamental trade-off here is that casual captures take much less time than fully controlled captures. However, this comes at the cost of a more difficult estimation of the camera path, e.g. by determining extrinsic calibration of video frames using SfM or SLAM.

Finding capturing procedures and flexible models to represent RealVR experiences is very exciting research since it combines many areas from computer vision and computer graphics.

5.2 Processing

Parallax360 does not rely on estimating camera extrinsics because of the fully controlled capture procedure using a robot arm. The core of its representation are curve-based disparity motion fields (see Figure 3), which are computed by processing disparity information obtained by optical flow between key frames and associated neighboring relative frames.

MegaParallax relies on the estimation of camera extrinsics. The ability to create datasets of sufficient quality converges with the problem to reconstruct scenes from a egocentric inside-out video. Once a camera path reconstruction of sufficient quality is obtained, it is straightforward to fit a circle to the estimated viewpoint centers, register viewpoints via polar angles induced by the circle, and to compute bidirectional optical flow between neighboring viewpoints (see Figure 5 Preprocessing).

Note that both methods demonstrate that optical flow works reliably if source and target images are *sufficiently* close to each other. An important remark is that the degree of closeness depends on the depth distribution of the scene. The closer scene objects come to a viewpoint, the bigger their disparity (motion parallax), when projecting them into a pair of neighboring frames or viewpoints.

While computing the curved motion fields can take up to 24 hours on a quad-core PC, and thus dominates the Parallax360 preprocessing time, SfM and optical flow computation takes less than 2 hours in MegaParallax.

5.3 Representation

Parallax360 computes the 5 parameters of an ellipse and 6 polar angles for every patch of every key frame to encode curve-based motion fields (see Figure 3). Key frames as well as novel viewpoints are defined *on* the spherical imaging surface that was sampled during capturing (see Figures 1 and 2). The viewing direction of a key or target frame is fully determined by the spherical coordinates modeling its central ray, i.e. a ray

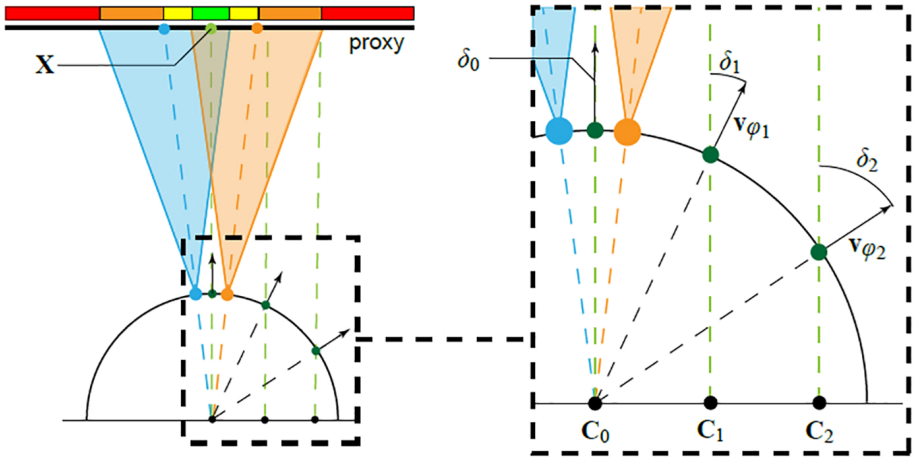

Fig. 11: MegaParallax viewing space analysis: **Left:** Blending performance according to central ray of the desired viewpoint, e.g. $\mathbf{C}_0$ placed at the center of the circle. **Right:** Translating the desired viewpoint ($\mathbf{C}_1$, $\mathbf{C}_2$) causes slanted angles with the circle, e.g. δ_1 and δ_2. If δ_i exceeds the field of view of the reference viewpoint, the ray reconstruction fails.

originating at the sphere's center and thus intersecting the spherical surface in normal direction (a.k.a. radial direction).

MegaParallax needs to estimate camera extrinsics per frame or viewpoint. Every viewpoint is registered with a circle fitted to the optical centers of cameras. For each pair of neighboring cameras, bidirectional flow is computed (see Figure 5, Preprocessing). In summary, for each key frame (viewpoint), a projection matrix is stored, as well as a polar angle and dense optical flow to its left and right neighbor.

The advantage of dense image representations is that no explicit scene geometry needs to be known to synthesize novel viewpoints. Nevertheless, this statement is not complete, since the quality of novel viewpoints depends heavily on the quality of correspondences between the images. This, in turn, varies with the nature of the scene to be captured. Note that visbility does not need to be modeled in dense image-based representations, view synthesis comes down to lookups and blendings and does not rely on reprojection over scene geometry.

5.4 Rendering

Synthesizing a novel viewpoint in Parallax360 (see Figure 4) is based on representing the target viewpoint with respect to its two or three closest key frames. The disparity motion fields of the key frames are then used to create intermediate target viewpoints by warping key frames *towards* the target viewpoint (see Figure 4 (b) and Equation 3). The target viewpoint is created by flow-based blending of the intermediate viewpoints, whose flow can be obtained from the disparity motion fields and the dense flow fields

between key frames (see Equation 4). Note that while the synthesis runs per pixel, the representation is based on image patches. Furthermore, novel viewpoints are always *interpolations* of existing pairs of key frames on a per-patch level.

View synthesis in MegaParallax (see Figures 6 and 8) is achieved by reconstructing individual camera rays, which leads to panoramic field of views and *extrapolation* of the captured viewpoints instead of only interpolation.

The most important aspect of flow-based blending is that *close* scene objects will project farther away from the centers of the source images, which introduces errors. This is very similar to the case of panorama stitching algorithms which mosaic vertical image strips into a single equirectangular image in 2D [14,21,15]. Nearby scene objects in the real world get *vertically distorted* [19] when packed into equirectangular images. They get literally *squeezed* horizontally which is desired in 360° panorama stitching algorithms, but unwanted in image-based rendering applications, in which correct perspectives in desired viewpoints are essential to immerse consumers into "RealVR" experiences.

5.5 Viewing space

The Parallax360 approach synthesizes novel viewpoints *on* the spherical imaging surface, which is sampled during the capture process. Novel viewpoints are synthesized using flow-based image interpolation. To enable head-motion parallax within the sphere, the initial target viewpoints need to be *extended*.

A viewpoint extension uses a homography and a scaling operation to adjust a synthesized viewpoint to approximate a desired viewpoint. The homography applies scaling to account for the expected change in parallax when moving forward or backward. Since this operation does not provide plausible motion parallax, we think that Parallax360 provides real head-motion parallax only on the surface of the sphere and not within. Nevertheless, the method supports arbitrary head-motions, but provides only restricted motion parallax when viewpoints need to be extended. The method employs a diameter of 1.25 m for the spherical imaging surface and shows compelling results for head-motion parallax in 360° environments. Nevertheless, the actual viewing space within the capturing sphere is not further evaluated.

MegaParallax operates *within* a circle, in comparison to Parallax360 which operates *on* the surface of a sphere. The translational freedom is restricted by (see Figure 11):

1. The radius of the circle,
2. the density of reference viewpoints and
3. their field of view shared with neighboring views.

The field of view of the desired viewpoint has an impact as well, but this parameter is kept fixed for each dataset. The actual viewing area is described by a concentric circle which supports high-quality view synthesis within a radius of roughly factor 2 of the capturing circle's radius depending on the quality of the extrinsic calibration (for more details, please refer to Bertel *et al.* [3], Section 8, Figure 14). As an example, hand-held datasets have a circle radius of roughly 0.8 m which leads to a viewing area radius of 0.4 m.

Fig. 12: Top row, Parallax360 limitation: Sequence of viewpoints with incorrect motion fields produce interpolation artefacts. Bottom row, MegaParallax limitation: The desired viewpoint is translated from the center of the circle to the left, outside the supported viewing space.

5.6 Limitations

The main limitations of Parallax360 are:

1. A time consuming capture and preprocessing stage and
2. view synthesis technically only on the surface of the imaging surface leading to view interpolation and incorrect motion parallax when moving inside the sphere.

Note that the quality of the view interpolation depends on the quality of the precomputed optical flow fields. The main limitations of MegaParallax are:

1. Only head-motion parallax inside the capture circle,
2. vertical distortion caused by using an inaccurate scene proxy and
3. fragile extrinsic estimation, which makes datasets from complex environments, such as specular or dynamic scenes or scenes with large depth variation, very challenging to reconstruct.

View synthesis fails in both approaches if a desired ray cannot be reconstructed from a determined pair of cameras (or triplet of key frames) due to insufficient field of view as discussed in Section 5.5. This results visually in black stripes as seen in the bottom row in Figure 12. From left to right: (1) The desired viewpoint placed at the center of the circle. (2) Viewpoint translates to the left. Camera rays on the right start retrieving black pixels because of slanted angles δ. (3) Translating further to the left causes more ray reconstructions to fail.

Another limitation shared by Parallax360 and MegaParallax is the dependency on sufficiently good flow fields. Optical flow can become unreliable in certain situations, e.g. when estimating correspondences in regions with repetitive textures (see Figure 12, top row).

It is worth noting that Parallax360 does not struggle with specular scenes because of the fully controlled capturing which is very similar in spirit to light field rendering [8]. Parallax360 does not suffer from vertical distortion since target viewpoints are created by blending warped key frames. The warping happens on the spherical surface and on a per patch-level within each key frame. Since *similar* perspective images are warped and blended, resulting viewpoints exhibit plausible perspectives. Note that this only works if the key frames are captured sufficiently dense with respect to the closest object[8] in the scene.

This is not the case for MegaParallax, which performs independent, per-ray reconstructions which can lead to cases in which a single viewpoint is synthesized by dozens of pairs of reference viewpoints which all show different perspectives, but cover a larger field of view. Since there is no accurate scene proxy to reproject reference viewpoints in 3D euclidean space, vertical distortion becomes visible in desired viewpoints (see how the hand-rail bends in Figure 8 of [3]).

6 Conclusion

Both Parallax360 and MegaParallax propose image-based scene representations and use implicit geometry, i.e. optical flow, to render novel viewpoints using a variant of flow-based blending. The absence of explicit geometry is compensated by a large number and high density of reference views, which are necessary to (1) compute flow fields reliably and (2) perform view interpolation while maintaining plausible perspectives. Both methods share time-consuming preprocessing stages, which enable real-time rendering algorithms based primarily on texture lookups and only light-weight computations at runtime.

The main advantage of the presented methods is the high visual quality of their results, particularly for the outdoor environments shown in this chapter (see Figures 9 and 10), which are known to be very difficult to reconstruct explicitly. Especially fine geometry like plants or trees are very challenging, but supported well in the results.

The main issues with MegaParallax are (1) vertical distortion that is caused by a constant scene depth assumption (as in concentric mosaics [19]) and (2) a viewing space restricted to a circle. The viewing space could be extended easily by using reference viewpoints with a wider field of view or providing explicit scene geometry to allow small off-plane movements and rotations. For Parallax360, the time-consuming capturing and processing stages as well as its viewpoint extension to render viewpoints within the sphere are its main issues.

While image-based rendering methods relying on explicit geometry produce the visually most compelling results for many environments, they rely on a successful estimation of scene geometry, which is hard to guarantee in arbitrary situations (see Figure

[8] Assuming a uniform distribution of reference viewpoints.

9 in [3]). The quality of the results mainly depends on the quality of the estimated geometry.

The interplay of capturing, processing, representation and rendering is vital to understand design decisions in "RealVR" pipelines. Since representations have to adopt to the nature of scene objects which shall be modeled, e.g. diffuse architecture vs. shiny cars, or hairy cats vs. thin twigs and leaves, more flexible representations have to be found to faithfully capture and represent the real world in all its beauty and visual fidelity.

To represent a real-world scene in all its complexity using only a sparse number of viewpoints, it seems natural to us that hybrid representations have to evolve, which combine the advantages of implicit and explicit geometry as well as learning-based approaches, such as multiplane images. The representations of the presented methods could be compressed by modeling static and diffuse parts of the scene with static geometry, since it is not necessary to store dozens of largely redundant images of a scene, which does not contain view-dependent effects.

We currently see the biggest demand in more robust and reliable methods for 2D and 3D scene understanding and scene reconstruction in particular. This will automatically provide better correspondences and thus more robust and more compact scene representations, which can be tweaked to perform excellently for special types of scenes. Finding a scene representation that is sufficiently flexible to model the real world has been a difficult challenge in computer graphics research for decades. Requiring that this representation should be *extractable* from a sparse set of viewpoints, ideally casually captured, makes novel-view synthesis for "RealVR" a very challenging and multidisciplinary research topic.

Acknowledgements

This work was supported by EU Horizon 2020 MSCA grant FIRE (665992), the EPSRC Centre for Doctoral Training in Digital Entertainment (EP/L016540/1), RCUK grant CAMERA (EP/M023281/1), the NSFC (No. 61671268, 61672307, 61727808), the National Key Technologies R&D Program of China (No. 2015BAF23B03), a UKRI Innovation Fellowship (EP/S001050/1), a Rabin Ezra Scholarship and an NVIDIA Corporation GPU Grant.

References

1. Adelson, E.H., Bergen, J.R.: The plenoptic function and the elements of early vision. In: Computational Models of Visual Processing, pp. 3–20. MIT Press (1991)
2. Anderson, R., Gallup, D., Barron, J.T., Kontkanen, J., Snavely, N., Hernandez, C., Agarwal, S., Seitz, S.M.: Jump: Virtual reality video. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **35**(6), 198:1–13 (November 2016). doi:10.1145/2980179.2980257
3. Bertel, T., Campbell, N.D.F., Richardt, C.: MegaParallax: Casual 360° panoramas with motion parallax. IEEE Transactions on Visualization and Computer Graphics **25**(5), 1828–1835 (May 2019). doi:10.1109/TVCG.2019.2898799
4. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). pp. 425–432 (2001). doi:10.1145/383259.383309

5. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). pp. 43–54 (August 1996). doi:10.1145/237170.237200
6. Hedman, P., Alsisan, S., Szeliski, R., Kopf, J.: Casual 3D photography. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **36**(6), 234:1–15 (November 2017). doi:10.1145/3130800.3130828
7. Hedman, P., Ritschel, T., Drettakis, G., Brostow, G.: Scalable inside-out image-based rendering. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **35**(6), 231:1–11 (November 2016). doi:10.1145/2980179.2982420
8. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). pp. 31–42 (August 1996). doi:10.1145/237170.237199
9. Luo, B., Xu, F., Richardt, C., Yong, J.H.: Parallax360: Stereoscopic 360° scene representation for head-motion parallax. IEEE Transactions on Visualization and Computer Graphics **24**(4), 1545–1553 (April 2018). doi:10.1109/TVCG.2018.2794071
10. McMillan, L., Bishop, G.: Plenoptic modeling: An image-based rendering system. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). pp. 39–46 (1995). doi:10.1145/218380.218398
11. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **38**(4), 29:1–14 (July 2019). doi:10.1145/3306346.3322980
12. Overbeck, R.S., Erickson, D., Evangelakos, D., Pharr, M., Debevec, P.: A system for acquiring, compressing, and rendering panoramic light field stills for virtual reality. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **37**(6), 197:1–15 (2018). doi:10.1145/3272127.3275031
13. Parra Pozo, A., Toksvig, M., Filiba Schrager, T., Hsu, J., Mathur, U., Sorkine-Hornung, A., Szeliski, R., Cabral, B.: An integrated 6DoF video camera and system design. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **38**(6), 216:1–16 (November 2019). doi:10.1145/3355089.3356555
14. Peleg, S., Ben-Ezra, M., Pritch, Y.: Omnistereo: Panoramic stereo imaging. IEEE Transactions on Pattern Analysis and Machine Intelligence **23**(3), 279–290 (2001). doi:10.1109/34.910880
15. Richardt, C., Pritch, Y., Zimmer, H., Sorkine-Hornung, A.: Megastereo: Constructing high-resolution stereo panoramas. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1256–1263 (2013). doi:10.1109/CVPR.2013.166
16. Schroers, C., Bazin, J.C., Sorkine-Hornung, A.: An omnistereoscopic video pipeline for capture and display of real-world VR. ACM Transactions on Graphics **37**(3), 37:1–13 (August 2018). doi:10.1145/3225150
17. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113 (2016). doi:10.1109/CVPR.2016.445
18. Shum, H.Y., Chan, S.C., Kang, S.B.: Image-Based Rendering. Springer (2007). doi:10.1007/978-0-387-32668-9
19. Shum, H.Y., He, L.W.: Rendering with concentric mosaics. In: Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH). pp. 299–306 (August 1999). doi:10.1145/311535.311573
20. Sweeney, C., Holynski, A., Curless, B., Seitz, S.M.: Structure from motion for panorama-style videos (2019), arXiv:1906.03539

21. Szeliski, R.: Image alignment and stitching: a tutorial. Foundations and Trends in Computer Graphics and Vision **2**(1), 1–104 (January 2006). doi:10.1561/0600000009
22. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. ACM Transactions on Graphics (Proceedings of SIGGRAPH) **37**(4), 65:1–12 (August 2018). doi:10.1145/3197517.3201323