An Omnistereoscopic Video Pipeline for Capture and Display of Real-World VR

CHRISTOPHER SCHROERS, Disney Research
JEAN-CHARLES BAZIN, KAIST
ALEXANDER SORKINE-HORNUNG*, Disney Research

In this paper we describe a complete pipeline for the capture and display of real-world Virtual Reality video content, based on the concept of omnistereoscopic panoramas. We address important practical and theoretical issues that have remained undiscussed in previous works. On the capture side we show how high quality omnistereo video can be generated from a sparse set of cameras (16 in our prototype array) instead of the hundreds of input views previously required. Despite the sparse number of input views, our approach allows for high quality, real-time virtual head motion, thereby providing an important additional cue for immersive depth perception compared to static stereoscopic video. We also provide an in-depth analysis of the required camera array geometry in order to meet specific stereoscopic output constraints, which is fundamental for achieving a plausible and fully controlled VR viewing experience. Finally, we describe additional insights on how to integrate omnistereo video panoramas with rendered CG content. We provide qualitative comparisons to alternative solutions, including depth-based view synthesis and the Facebook Surround 360 system. In summary, this paper provides a first complete guide and analysis for reimplementing a system for capturing and displaying real-world VR, which we demonstrate on several real-world examples captured with our prototype.

CCS Concepts: • **Computing methodologies** → **Computational photography**;

Additional Key Words and Phrases: virtual reality, omnidirectional videos, stereoscopy, real-world content.

ACM Reference Format:
Christopher Schroers, Jean-Charles Bazin, and Alexander Sorkine-Hornung. 2018. An Omnistereoscopic Video Pipeline for Capture and Display of Real-World VR. *ACM Trans. Graph.* XX, XX, Article XX (2018), 13 pages. https://doi.org/0000001.0000001_2

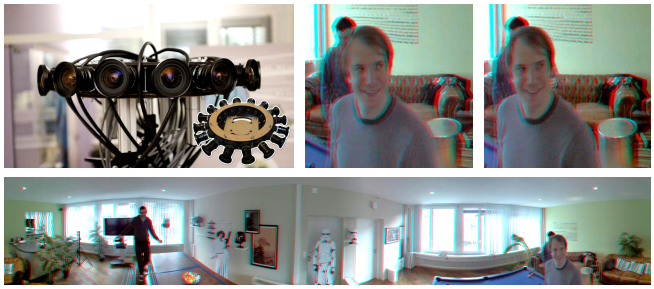

Fig. 1. Given videos acquired by a sparse camera rig (see our prototype with 16 cameras on the left), we describe a complete pipeline for the generation of high-quality stereoscopic omnidirectional videos of dynamic scenes (bottom), with support for smooth parallax interpolation between different viewpoints (two top right images), enabling real-time virtual head motion in Virtual Reality applications despite the sparse video input.

1 INTRODUCTION

Technologies for Virtual and Augmented Reality are currently experiencing a new boom. Companies such as Facebook (Surround 360), Google (Jump), Jaunt VR, GoPro (Omni), Samsung (Gear VR), Microsoft (Hololens), Magic Leap, Oculus, and many others are marketing omnidirectional camera systems and head-mounted displays. While the basic concepts underlying Virtual Reality (VR) systems have been around already for a few decades, the performance of hardware and software for capture, rendering, and display is now getting to a point where more immersive experiences are possible.

While rendered VR based on computer generated content has reached an impressive level, the capture of high resolution, immersive stereoscopic real-world content remains challenging in practice, since it usually has to be stitched from multiple input video streams, e.g., using camera arrays. Seamless 2D stitching of such content is nowadays supported by many camera systems. However, in order to capture and display high quality *stereoscopic 3D* content, many more views with larger inter-camera parallax have to be captured, and one has to convert the captured parallax into consistent, seamless stereoscopic output images.

One potential approach is image-based rendering utilizing depth information extracted from the input views (e.g., [Shum and Kang 2000]), where the stereoscopic output is generated depending on the viewer's viewing position and direction. However, with only sparse input available as in our setting, robust and sufficiently accurate depth estimation for arbitrary scenes remains challenging, often resulting in noticeable visual artifacts.

A milestone in this context have been the works of Peleg et al. [1999; 2001], which demonstrated how view-direction *independent* images can be generated from densely captured images using a single camera rotating off-center. This insight significantly simplified the creation of panoramic stereo, since a single pair of correspondingly generated panoramic images can provide a plausible omnistereoscopic impression, i.e. a stereoscopic impression in all viewing directions on the equatorial plane. The price is that the generated images are multi-perspective, and hence do not exactly correspond to a real-world, view-direction dependent stereo geometry that is identical to human stereo perception.

A key advantage of their omnistereoscopic representation in the context of VR is that only a small modification to the stitching process allows for changes of the virtual viewpoint position, enabling a more natural interaction and immersive experience of the VR

*Alexander is now at Oculus. This work was completed during his time at Disney Research.

content compared to static stereoscopic images. However, in practice, existing high resolution omnistereo approaches (e.g., [Richardt et al. 2013]) require a very large number of input views, in the order of several tens to hundreds, to generate convincing virtual head motion effects. Building a corresponding video camera array is a difficult practical challenge.

In this paper we describe a complete and practicable pipeline for the capture and display of real-world VR video content. In particular we try to address important questions that have remained open in previous works. On the practical side, we discuss a solution that allows the generation of plausible stereo panoramas, including virtual head motion, from a small number of input videos. Our prototype is a camera array consisting of only 16 machine vision cameras (see Fig. 1). On the theoretical side, we provide an analysis that relates stereoscopic output parallax and the available virtual head motion to the camera array geometry, which is an important prerequisite in order to be able to capture convincing, immersive real-world VR. We further discuss additional tools such as multi-perspective projection in order to mix real-world VR with computer generated content.

We show several real-world video panoramas that are suitable for real-time stereoscopic viewing with support for virtual head-motion, e.g., using an Oculus or Google Cardboard HMD. We compare to alternative solutions such as image-based rendering using depth, and to the Facebook Surround 360 system. In addition, we discuss and simulate alternative array and camera designs that, we hope, provide a useful guide for other researchers for building omnistereo video capture systems.

2 RELATED WORK

2D panoramas of a scene are traditionally obtained by stitching multiple pictures sharing a common single center of projection, e.g. acquired by a purely rotating perspective camera [Brown and Lowe 2007; Hartley and Zisserman 2004; Kopf et al. 2007]. See the extensive survey by Szeliski [2006] for more details. 2D panoramic stitching tools are now commonly available on consumer cameras and smartphones. Similar tools and products also exist for video panoramas, where multiple cameras with overlapping field of view capture the scene, such as the Point Grey Ladybug camera, the FlyCam, or the PanaCast camera. The issue with such camera arrays is that, in order to create seamless 2D output panoramas, one has to hide or compensate for the parallax between the individual input views, which can for example be achieved by image warping [Jia and Tang 2008; Kang et al. 2004; Lee et al. 2016; Perazzi et al. 2015; Shum and Szeliski 2000] or seam-optimization [Efros and Freeman 2001; Zhang and Liu 2014]. See, e.g., Perazzi et al. [2015] for a more detailed overview of 2D approaches.

In contrast to 2D panoramas, for the creation of stereoscopic panoramas, parallax between input views is desired, i.e., the scene has to be observed from different viewpoints. One solution is to rotate a pair of stereo cameras [Couture et al. 2011; Zhang and Liu 2015] or a lightfield camera [Birklbauer and Bimber 2014] and then stitch the acquired pictures. These approaches provide limited stereo due to the short camera baseline, and are challenging to extend to omnidirectional stereoscopic video acquisition for dynamic scenes as required for VR applications. [Hedman et al. 2017] capture images from a moving camera, and then perform textured 3D mesh reconstruction that can be used for view synthesis. While this allows a wider stereo effect, it is also limited to static scenes. Another alternative is pushbroom imaging, which has been extensively applied and studied in the context of satellite images [Gupta and Hartley 1997], and for linear acquisition, e.g., of facades [Agarwala et al. 2006; Kopf et al. 2010; Rav-Acha et al. 2008; Román and Lensch 2006; Zheng et al. 2011]. Such approaches still do not support dynamic scenes and 360 omnidirectional stereo.

To overcome these limitations, it was proposed to rotate a camera off-center [Ishiguro et al. 1992; Peleg and Ben-Ezra 1999; Peleg et al. 2001; Richardt et al. 2013; Shum and He 1999]. Stereoscopic panoramic images can then be created by selecting appropriate columns from the input images. A key aspect of these approaches is that the resulting stereo panoramas are omnidirectional, i.e., a single pair of panoramic images generated with these approaches provides a plausible stereo impression in all viewing directions on the equatorial plane. While the resulting images are multi-perspective and hence are not equivalent to the view-direction dependent result of, e.g., a regular stereo camera pair, the resulting stereo is still visually convincing (see [Seitz and Kim 2002] for a detailed discussion). An important advantage in the context of our work is that such a view-direction independent solution is much more feasible in the context of real-time VR applications, because a single pair of images can be used to provide stereo in all viewing directions on the equatorial plane, leading to significantly reduced computational overhead and storage requirements [Shum et al. 2005; Simon et al. 2004]. Recently it has been shown that omnistereoscopic panoramas can be created using as few as three extreme wide angle lenses [Chapdelaine-Couture and Roy 2013] or only two 360 spherical cameras [Matzen et al. 2017]. However this is at the price of considerably reduced effective output resolution. A second major advantage of this omnidirectional representation is that it rather easily supports changes in perspective, corresponding to a virtual head motion of the viewer, without the need for more complex image-based rendering techniques requiring an accurate depth representation [Shum and Kang 2000].

A remaining limitation of the above approaches is, however, that video results of dynamic real-world scenes are challenging to create in practice. The main reason for this limitation is that all above approaches require a comparatively large number of input views to capture a sufficiently dense lightfield of the scene, which makes it impossible to capture dynamic content data with an array of cameras. A few companies have recently announced commercial solutions based on camera rigs similar to ours (e.g., Facebook Surround 360, Google Jump, Jaunt VR). While the Google Jump algorithm is described in [Anderson et al. 2016], the underlying methods are proprietary and not accessible. Only Facebook Surround 360 provides publicly available source code. In the experiments section we provide qualitative comparisons to the Facebook system.

In summary, we describe in detail how a sparse camera rig can be used in order to create high quality omnistereo, including support for virtual head motion and fully dynamic scenes, making capture and VR display of real-world scenes possible. In addition we discuss all necessary system parameters in order to gain full control over the

resulting output stereo, which is essential for creating immersive virtual experiences.

3 OVERVIEW

First, we will describe how to process the input views coming from a sparse omnidirectional camera array in order to create the dense representation required for synthesizing omnistereoscopic panoramas with support for virtual head motion. We discuss how to parameterize and interpolate the captured light rays using pure image-domain operations, so that despite the sparse input, no accurate scene information (such as depth) is required.

In the second main section, we then provide a detailed analysis of various aspects of this pipeline, ranging from expected stereoscopic output parallax to the range of virtual head motion that is feasible for a particular array design. We provide derivations for ray tracing and projection of computer generated content directly into the multiperspective output panoramas, and discuss various additional aspects relevant for using the proposed pipeline in practical VR applications.

4 PANORAMA CREATION FROM SPARSE INPUT

4.1 Camera setup and parameterization

We consider a sparse camera array composed of n cameras covering a complete 360-degree field of view, see Fig. 1. In a preprocess we estimate the camera calibration parameters (extrinsic and intrinsic) with standard calibration techniques [Zhang 2000] and bundle adjustment, and then radially undistort the images.

In an ideal setup the cameras lie on a circle of radius r, and the camera intrinsic calibration matrix can be written

$$\mathsf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where f_x and f_y denote the focal length in pixels in the horizontal and vertical directions, and (c_x, c_y) the principal point [Hartley and Zisserman 2004]. Noting α the angle of the camera along the circle, the pose of each camera is defined as

$$\mathsf{R}(\alpha) = \begin{bmatrix} -\sin(\alpha) & 0 & -\cos(\alpha) \\ 0 & 1 & 0 \\ \cos(\alpha) & 0 & -\sin(\alpha) \end{bmatrix} \text{ and } \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix}. \tag{2}$$

The corresponding camera projection matrix that maps 3D world coordinates to pixel coordinates is

$$\mathsf{P}(\alpha) = \mathsf{K}\,[\,\mathsf{R}(\alpha) \mid \mathbf{t}\,]\,. \tag{3}$$

We then define the dense lightfield for a single point in time as $L(\alpha, x, y)$, which we have to reconstruct from the sparse camera input (see Fig. 2 for an example of a synthetic scene). The value at a location (α, x, y) corresponds to the measured irradiance along the ray

$$\mathsf{R}^\top(\alpha)\left(\lambda \cdot \mathsf{K}^{-1}(x, y, 1)^\top - \mathbf{t}\right), \tag{4}$$

where $\lambda > 0$ represents the position along the ray.

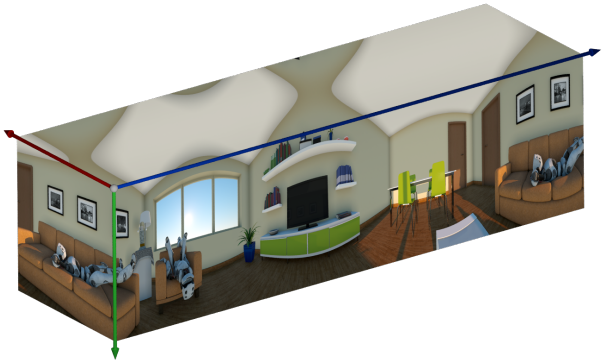

Fig. 2. Dense lightfield representation of a synthetic scene. The x, y and α axes are shown in red, green and blue, respectively.

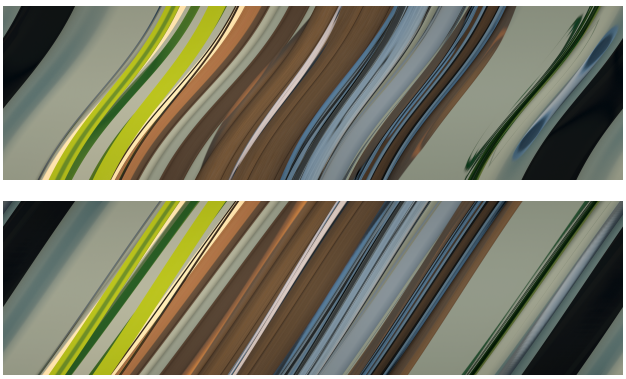

Fig. 3. Top: α-x slice of $L(\alpha, x, y)$ from Fig. 2. Bottom: α-x slice after transforming with the mapping function φ. The fact that after transformation the curved lines are straightened illustrates that a linear interpolation in the transformed space can result in accurate point trajectories.

4.2 Lightfield Reconstruction

Our aim is to reconstruct $L(\alpha, x, y)$ from a sparse set of input views $\hat{I}_i$ with $i = 1, \ldots, n$. The first step is to map the captured input images into the coordinate frame of L by associating each input image $\hat{I}_i$ with a camera angle α_i using the estimated camera calibration. In order to approximate the ideal camera setup described in the previous section, where all cameras reside on a circle, we align each input image $\hat{I}_i$ with the expected input at angle α_i by applying the corresponding homography. Referring to the thus transformed images as I_i, we now have

$$L(\alpha_i, x, y) \approx I_i(x, y). \tag{5}$$

Then the problem of reconstructing L comes down to performing an accurate view interpolation in α. For an accurate image space approach, it is important to understand how a given 3D point $\mathbf{X}$ moves in (x, y) when varying the camera angle α.

Trajectories of scene points in image space. Rather than considering the projection of a fixed 3D point when rotating the camera about the origin $\mathbf{o}$ by some angle α, changing the point of view

Fig. 4. Transforming an input image with φ yields a good approximation to the point trajectories even for points that are close. This can be illustrated by the fact that varying the cylinder radius d used in φ_d from 1 to ∞ interpolates between the input image and the image transformed by φ. Unless points are unreasonably close for typical capturing setups, i.e., $d \ll 2r$, the resulting transformations closely resemble each other. This is best visible when comparing the shapes of the transformed images.

can provide a more intuitive understanding: by keeping the camera fixed and rotating the 3D point with the inverse rotation instead, the same trajectory can be obtained. The path can thus be interpreted as observing a 3D point that travels along a cylindrical surface.

Assuming that the depth at a given location $\mathbf{x} = (x, y)^\top$ is known, the nonlinear path in image space can be reconstructed by backprojecting according to Eq. 4, rotating the resulting 3D point $\mathbf{X}$, and finally projecting it with Eq. 3*. When representing the 3D point $\mathbf{X}$ in cylindrical coordinates, its change in position is linear in the angle α. Let the map $\varphi_d : \mathbb{R}^2 \to \mathbb{R}^2$ denote the backprojection onto a cylinder with radius d followed by a conversion to cylindrical coordinates. Then knowing two corresponding image points $\mathbf{x}_i$ and $\mathbf{x}_j$ measured at angles α_i and α_j and their radial depth d w.r.t. to the origin $\boldsymbol{o}$ allows to define the nonlinear path in image space

$$\mathbf{x}(\alpha) = \varphi_d^{-1}\Big((1 - t(\alpha)) \cdot \varphi_d(\mathbf{x}_i) + t(\alpha) \cdot \varphi_d(\mathbf{x}_j)\Big) \tag{6}$$

in terms of a linear interpolation in the transformed space. Here we assume that $\alpha_i < \alpha < \alpha_j$ such that the weight $t(\alpha)$ is given by

$$t(\alpha) = \frac{\alpha - \alpha_i}{\alpha_j - \alpha_i}. \tag{7}$$

Although φ_d does allow using image space correspondences for an accurate view interpolation, it still depends on the depth of the scene point as it determines the radius of the cylinder. However, Fig. 4 illustrates that the transformation an image undergoes is almost constant when varying the cylinder radius from around $2r$ to ∞. This indicates that trajectories of points can be well approximated by using a very large cylinder radius, even when they are relatively close. By letting $d \to \infty$, one can express the mapping in a depth independent way as

$$\varphi(\mathbf{x}) = \begin{pmatrix} \omega(\mathbf{x}) \\ s(\mathbf{x}) \end{pmatrix} \tag{8}$$

with

$$\omega(\mathbf{x}) = \operatorname{atan}\left(\frac{x - c_x}{f_x}\right) \quad \text{and} \quad s(\mathbf{x}) = (y - c_y) \cdot \cos(\omega(\mathbf{x})). \tag{9}$$

* This idea can also be extended to use other camera models, such as fisheye projection models, potentially with a field of view larger than 180°, by adapting the equations for backprojection and projection as desired.

This is straightforward considering that $d \to \infty$ is equivalent to letting the camera circle radius $r \to 0$.

Fig. 3 depicts α-x-slices of $L(\alpha, x, y)$ before and after transformation with φ. Curved lines become straightened after the transformation which indicates that linear interpolation indeed is an appropriate approximation to the point trajectory. Due to this insight, we are now able to compute intermediate views based on image space correspondences as follows.

Computing intermediate views. As a preprocessing step, we first compute forward and backward flows between all consecutive image pairs. To this end, we slightly adapt a commonly available method [Brox et al. 2004] and minimize the energy

$$\begin{aligned} E(\mathbf{u}_{ij}) = & \int_\Omega \Psi\left(|I_i(\mathbf{x}) - I_j(H_{ij}(\mathbf{x} + \mathbf{u}_{ij}(\mathbf{x})))|^2\right) \, \mathrm{d}\mathbf{x} \\ & + \gamma \int_\Omega \Psi\left(|\nabla I_i(\mathbf{x}) - \nabla I_j(H_{ij}(\mathbf{x} + \mathbf{u}_{ij}(\mathbf{x})))|^2\right) \, \mathrm{d}\mathbf{x} \\ & + \beta \int_\Omega \Psi\left(|\mathcal{J}\mathbf{u}_{ij}(\mathbf{x})|^2\right) \, \mathrm{d}\mathbf{x} \, . \end{aligned} \tag{10}$$

As it is commonly done, we use a robust penalization function $\Psi(s^2) = \sqrt{s^2 + \varepsilon^2}$, where $\varepsilon = 10^{-3}$ is a small constant and $\mathcal{J}$ denotes the Jacobian. However, instead of directly computing correspondences between the original input images I_i and I_j we propose to leverage the cameras intrinsic and extrinsic calibration information to guide the flow estimation. To achieve this, we effectively preregister I_j to I_i using the homography induced by the plane at infinity $H_{ij} = K_j R_{ij} K_i^{-1}$ [Hartley and Zisserman 2004]. Incorporating the homography H_{ij} into the minimization problem allows for a better initialization since accounting for the camera intrinsics and the relative rotation between the cameras allows to bring all objects into closer alignment in image space. In fact, distant objects can already be well aligned by this homography such that the correspondence estimation problem now mostly comes down to refining initial matches on closer objects. Since optical flow estimation is a non-convex problem, having a better initialization is important as it makes it less likely to get stuck in local minima. Besides yielding a better initialization, incorporating the homography into the

minimization problem offers an additional advantage: since a homography can describe a nonlinear mapping in image space, it can account for some of the linear and nonlinear parts of the overall correspondence field between the original input images I_i and I_j which otherwise may be hard to reconstruct with a first order smoothness term that pushes the correspondence field to be piecewise constant.

Initially we anticipated a temporal consistency term in the flow estimation would be required to tackle temporal flickering. However, in practice, we observed that the results obtained by our independently computed flows are temporally sufficiently consistent (see supplementary video). The same observation has also been made by [Perazzi et al. 2015] for monoscopic panoramic video stitching.

Following the findings of the previous section, we now use these correspondences to synthesize intermediate views. When given a camera angle $\alpha \in [0, 2\pi)$, we can find the two closest input images which are related to the cameras capturing views at the angles α_i and α_j, respectively. Then we compute the warp fields

$$\begin{aligned} \mathbf{w}_{ij} &= \varphi^{-1}\Big((1-t(\alpha))\cdot\varphi + t(\alpha)\cdot(\varphi\circ H_{ij}\circ(\mathrm{id}+\mathbf{u}_{ij}))\Big) \\ \mathbf{w}_{ji} &= \varphi^{-1}\Big(t(\alpha)\cdot\varphi + (1-t(\alpha))\cdot(\varphi\circ H_{ji}\circ(\mathrm{id}+\mathbf{u}_{ji}))\Big) \end{aligned} \tag{11}$$

and synthesize the novel view from angle α as

$$L(\alpha, x, y) = (1-t(\alpha))\cdot I_i^{\mathbf{w}_{ij}}(\mathbf{x}) + t(\alpha)\cdot I_j^{\mathbf{w}_{ji}}(\mathbf{x}), \tag{12}$$

where $I_i^{\mathbf{w}_{ij}}$ corresponds to I_i forward-warped using $\mathbf{w}_{ij}$. A single panoramic image can then be obtained by fixing x, i.e., a particular image column, to obtain an α-y slice of L (see Fig. 2). Correspondingly, a stereoscopic output panorama can be created by picking two α-y slices at different column positions x.

Usually, it is desirable to have square pixels in the output panorama. Therefore we determine the sampling rate in α such that the pixel width in the output panorama matches the pixel height in the input image. The pixel height in the input image corresponds to $1/f_y$ when considering an image plane at distance 1. With n samples, the pixel size in the output panorama is $2\pi/n$. Thus we use $n = 2\pi f_y$ samples in α when aiming for square pixels.

This concludes the explanation of our interpolation method. Not only the choice of the method, but also the choice of the interpolation points is important. In this scenario the choice of interpolation points corresponds to the camera setup which we detail on next.

5 ANALYSIS AND PRACTICAL GUIDELINES

In order to create perceptually plausible stereoscopic video, e.g., for VR applications using head-mounted displays, it is essential to have full control over the output parameters such as parallax, stereoscopic disparity, or the possible amount of virtual viewpoint changes in the panoramic output images.

We therefore first have to understand the image formation model, i.e., how 3D points are projected into a panorama. Knowing this, we can measure the role of different parameters of the capture system such as the number of cameras n, the circle radius r and the focal lengths f_x and f_y, on a given panorama.

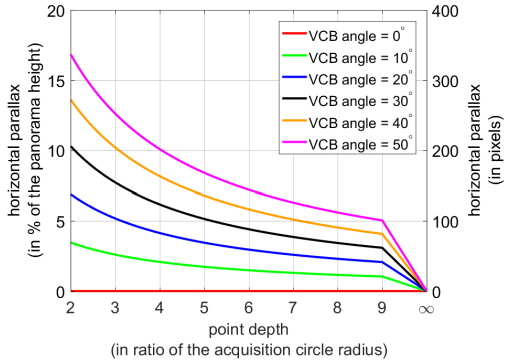

Fig. 5. Evolution of the horizontal parallax of a scene point in the output panorama with respect to its depth for different virtual camera baseline (VCB) angles. The parallax is expressed in percentage of the panorama height (left y-axis) and in pixels (right y-axis). The input images have a resolution of 2000×2000 pixels with 80° field of view.

5.1 Panoramic image formation

When fixing α, the projection of Eq. 3 allows to project a world point $\mathbf{X} = (X, Y, Z)^\top$ by

$$\begin{pmatrix} x \\ y \end{pmatrix} \cong \mathsf{K}(\mathsf{R}(\alpha)\mathbf{X} + \mathbf{t}) \tag{13}$$

where (x, y) is the projection in inhomogenous coordinates (by the operator $\cong$). Since a panorama is just a (α, y) slice of L obtained by fixing x, the panoramic camera model can be obtained by fixing x instead of α and solving for (α, y). This leads to an equation of the form (see details in the Appendix)

$$A\sin(\alpha) + B\cos(\alpha) = C \tag{14}$$

with coefficients

$$A = X\cdot f_x - Z\cdot(x - c_x) \tag{15a}$$
$$B = Z\cdot f_x + X\cdot(x - c_x) \tag{15b}$$
$$C = -r\cdot(x - c_x) \tag{15c}$$

Two solutions exist:

$$\alpha_1 = \phi - \gamma \tag{16a}$$
$$\alpha_2 = \pi - \phi - \gamma \tag{16b}$$

up to 2π, where $\phi = \sin^{-1}\left(\frac{C}{D}\right)$, $D = \sqrt{A^2 + B^2}$ and $\gamma = \tan^{-1}\left(\frac{B}{A}\right)$. To obtain α, we simply pick the solution for which $\mathbf{X}$ lies in front of the camera. Given α it is straightforward to obtain y by the projection of Eq. 13.

5.2 Amount of stereoscopic parallax

With the panoramic image formation model we can now understand the relation between the scene depth and the parallax in the output panoramas. As explained in Sec. 4.2, the stereoscopic output panorama is created from two column slices. For example, the left panorama is created from the slice at column x_l, and the right panorama at column x_r. For simplicity, in this section, we assume that all cameras in the rig have a fixed focal length f. Furthermore, we consider symmetric cases around the center column x_c of the

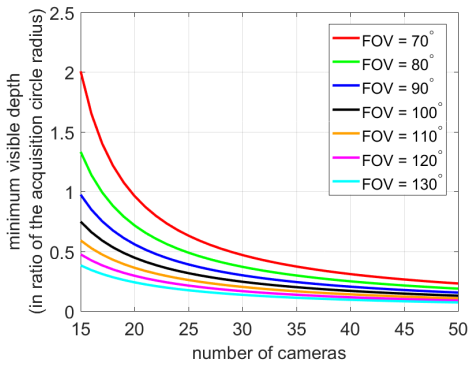

Fig. 6. Evolution of the minimum depth visible (distance from the acquisition circle) in two cameras with respect to the number of cameras on the rig and with different fields of view.

input images, i.e., $|x_l - x_c| = |x_r - x_c|$. The distance $x_r - x_l$ controls the virtual camera baseline (VCB). This is analogous to the distance between a pair of cameras in a conventional stereo rig controlling the resulting stereoscopic output parallax. To conduct experiments in a way that is invariant to the input image size, we consider the VCB angle which is given by $2 \cdot \omega(x_r)$ according to Eq. 9.

Fig. 5 verifies that the parallax in the stereoscopic output panorama decreases with larger scene depth and smaller VCB angle. Furthermore, it also shows how to choose the acquisition and synthesis parameters to match the disparity capabilities of desired output devices, such as head mounted displays or autostereoscopic screens with a limited disparity range.

5.3 Minimum visible depth

In order to capture a large amount of parallax, objects have to be sufficiently close to the camera (Sec. 5.2). However, due to the sparse sampling, an object too close to the camera array might be observed by only one camera, and therefore cannot be correctly interpolated. An important question that naturally arises is: what is the minimum distance that can be observed by two cameras. We derive that the minimum distance is (see details in the Appendix)

$$r \frac{\sin(\pi - \beta/2)}{\sin(\beta/2 - \theta)}, \tag{17}$$

where r is the camera circle radius, β is the field of view of the camera, and $\theta = 2\pi/n$ is the angle between two cameras (uniform distribution on the camera rig circle). Fig. 6 illustrates the values of the minimum depth with respect to the number of cameras of the rig and the camera's field of view. It shows that the minimum distance decreases with more cameras and wider field of view.

5.4 Analysis of the image space view interpolation

In Sec. 4.2 we mentioned that the trajectories of scene points can be well approximated solely by image space correspondences without knowing their actual depth. This section analyzes the approximation quality of the image space view interpolation in detail and shows that the deviation compared to a view interpolation based on scene depth is small. Our analysis covers different configurations of camera arrays as well as varying depth and elevation of world points. Here, the elevation of a world point is defined as the angle between the camera rig plane and the line connecting the world point and the camera rig center. The plots in Fig. 7 show that the interpolation error is small and goes towards zero when the world point is far away from the cameras, the field of view is wider (Fig. 7a) and more cameras are used (Fig. 7b).

The above experiments, in conjunction with Fig. 6, indicate that it is benefical to use a camera array composed of many cameras with a wide field of view. On the other hand, using many cameras increases the requirements in terms of data bandwidth, memory, synchronization and processing. A wider field of view decreases the effective resolution. Concretely, this means that users need to carefully consider the trade-off between the scene volume that can be captured and the setup requirements; or adapt the acquisition setup according to the data provided in Fig. 6 and Fig. 7.

5.5 Virtual head motion

A particularly intriguing feature of the omnistereoscopic panorama representation is the ability to simulate virtual head motion, i.e., shifting the viewer's location sideways within the captured scene. In practice, sideways headmotion can be achieved simply by synthesizing the stereoscopic output panorama using two columns from the lightfield L that are not symmetrically placed around the center column, which in turn provides a view of the scene from varying perspectives. This principle has been demonstrated, for example, in [Peleg et al. 2001; Richardt et al. 2013]. However, in order to be applicable in real-time VR applications, e.g., using a head-tracked Oculus Rift, a user's head motion has to be properly mapped.

One issue is that picking a α-y slice from the lightfield L for generating a panorama (Sec. 4.2) not only changes the perspective onto the scene, but also modifies the orientation of the panorama (see Fig. 8). In order to synthesize proper output panoramas required for virtual head motion effects, the orientation between the panoramas must stay consistent. This means that points at infinity must be fixed in the generated panoramas, i.e., be at the same location. Let $\mathcal{P}$ and $\mathcal{P}'$ be two panoramas generated from extracting α-y slices from L by fixing the columns x and x', respectively. Noting α and α' the orientation of a point at infinity in $\mathcal{P}$ and $\mathcal{P}'$, we must then have $\alpha = \alpha'$. In practice, this can be achieved by shifting $\mathcal{P}'$ by $\omega(x') - \omega(x)$ where $\omega(x)$ is defined in Eq. 9 (see details in the Appendix). Fig. 8 illustrates the effect of fixing points at infinity on a synthetic sequence.

After this registration, a virtual head motion effect that mimics a sideways head motion can be achieved by tracking the sideways headmotion of a user and selecting the panorama based on this data. In our prototype, we typically use a subset (just 10-20) of all panoramic images as this is sufficient for creating a convincing head motion effect. These views are precomputed and then loaded by the VR viewer in real-time. This approach directly transfers to the stereoscopic case where one simply selects both the left and the right panoramas based on the head position. The described approach allows for a real-time head motion effect in stereo as it comes down to selecting two appropriate panoramas. Although the head motion parallax induced by the sideways head motion is

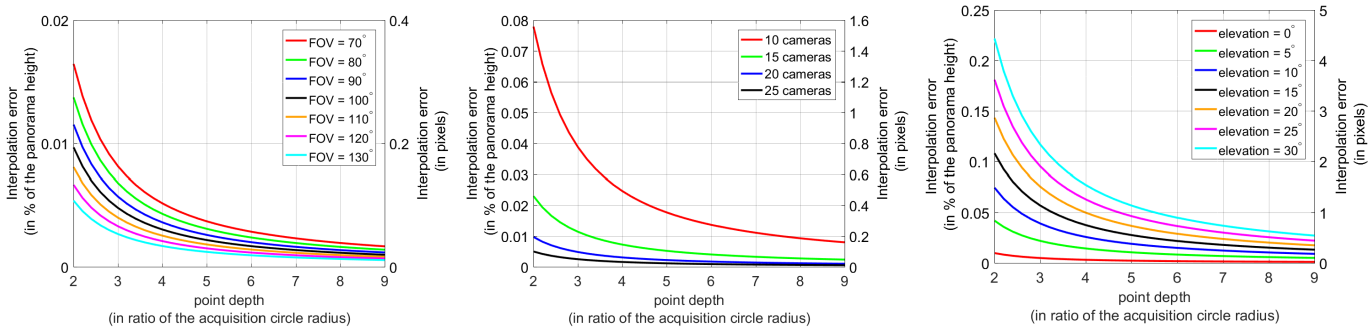

Fig. 7. Interpolation error in percentage of the panorama height (left y-axis) and in pixels (right y-axis) from input images with a resolution of 2000×2000 pixels. (a) interpolation error w.r.t. varying field of view (FOV) for an array composed of 20 cameras, and a world point elevation of 0°. (b) interpolation error with varying numbers of cameras, FOV=100°, and a world point elevation of 0°. (c) interpolation error with varying elevations of the world point, 20 cameras and FOV=100°.

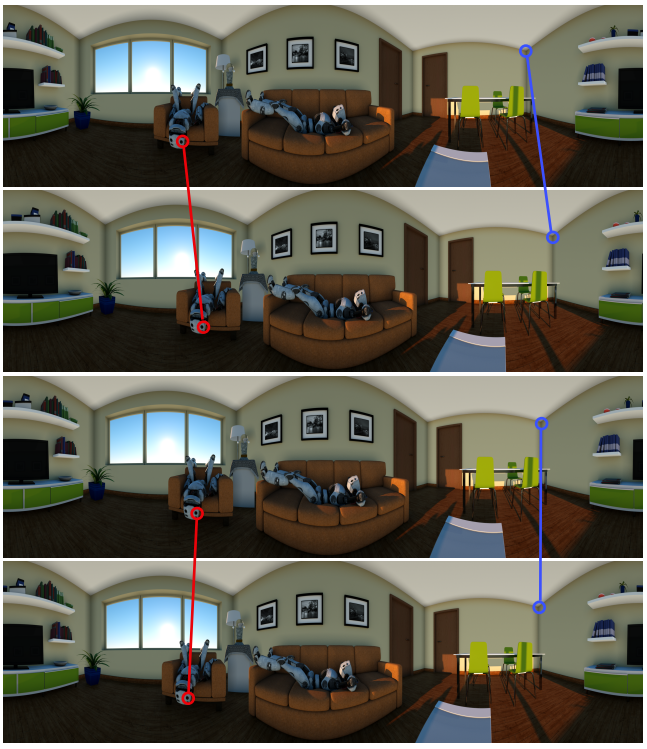

Fig. 8. Example of two output panoramas $\mathcal{P}$ and $\mathcal{P}'$ without (top two) and with (bottom two) fixing the panorama orientation. Fixing the orientation results in the expected behaviour of closer points (red) exhibiting stronger motion parallax than far points (blue).

usually best observed when the user is not rotating at the same time, this is not strictly required. Our system also supports the scenario where sideways head motion and rotation occur simultaneously. Results of head motion with fixed orientation in stereo are available in the supplementary video.

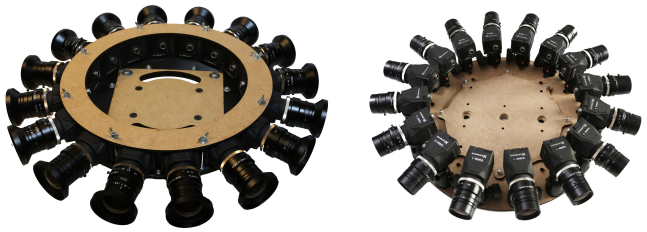

Fig. 9. Our two camera rig prototypes used for capturing panoramic content. The one on the left with a 3D printed camera mount has more accurate orientation and position of the cameras and lenses with a wider field of view compared to the hand-assembled rig on the right.

6 RESULTS

In this section we discuss results generated with our prototype array and the described processing pipeline. Please see the supplemental material for dynamic video results.

Real world results. We show real world results for different scenarios including static setups where the camera rig is fixed as well as dynamic setups with a moving camera array, see Fig. 1 and Fig. 13. Our experiments cover both indoor and outdoor scenes. For capture, we have used two different 360° camera rig prototypes. One has been assembled by hand with approximate camera placement and orientation, while the other one has been created with a 3D printer (Fig. 9). Both prototypes are equipped with $n = 16$ synchronized cameras that are more or less uniformly distributed on a circle of radius r =20cm and r =15cm, respectively, with cameras pointing outwards.

We used Lumenara Lt425 cameras equipped with Kowa lenses with a field of view of approximately 70° and 50°, respectively, and captured videos at a resolution of 2048 × 2048 pixels per camera and 30 frames per second.

We processed the acquired videos on a standard desktop computer equipped with an Intel i7 3.2Ghz and 64GB RAM. Our current C++

Fig. 10. Top: Resulting panorama without using flow, i.e., setting all $\mathbf{u}_{ij} = \mathbf{0}$. Bottom: Resulting panorama when using flow.

Fig. 11. Reconstructed panoramas without (top) and with (bottom) considering the extrinsic and intrinsic camera parameters. Note the significant, spatially varying distortion of the scene in the top image, such as the car and window. A consistent result as in the bottom is crucial, in particular for dynamic scenes and for plausible virtual head motion effects.

prototype implementation running on CPU takes about 20 minutes per time instance in total from the reading of the n input images to the generation of all the panoramas with the complete range of column disparities, including optical flow computation, which is the current bottleneck. This is an offline preprocess that does not require any user interaction. Once the panoramas are generated, our view-independent approach allows real-time head-motion effects. We expect significant speed improvements from an optimized GPU implementation.

Necessity of flow-based correspondences. Fig. 10 shows crops of panoramas with and without using the flow-based correspondences. The latter one is achieved by simply setting all flows $\mathbf{u}_{ij} = 0$ during panorama reconstruction. In this case ghosting artifacts occur because homographies alone do not allow for an accurate reconstruction. Without using an accurate interpolation, creation of stereoscopic content and changes of viewpoint are not possible. In general, if the optical flow is inaccurate, artifacts may become visible in the output panorama. This is not specific to our method, but common for all flow-based techniques. Both Fig. 10 and Fig. 19 can give an intuition of how our system behaves when supplied with inaccurate optical flow. Given that they constitute extreme cases and still the generated panoramic image is well recognizable, this shows that our system, in particular due to our pre-alignment, is designed to fail gracefully when supplied with inaccurate optical flow.

Fig. 12. Top: Head-motion effect on a close-up view of the indoor panorama. Note the significant parallax between the two images, especially the foreground person occluding the body of the background person and the metallic stool. See the video for a smooth interpolation. Bottom: Head-motion effect on a close-up view of an outdoor panorama.

Influence of calibration. Fig. 11 illustrates the errors that occur when neglecting the camera parameters in the reconstruction process (Sec. 4.2). Especially for our handmade prototype, the real camera pose can considerably deviate from the theoretically expected one. In this case, the flow-based interpolation will automatically compensate for these deviations and still produce a photometrically consistent panorama, e.g. without artifacts such as ghosting. However, this comes at the cost of distorting the scene geometry. Considering the camera parameters allows to produce geometrically consistent panoramas which is crucial for creating stereoscopic content and viewpoint animations.

Virtual head motion. Our method produces accurate reconstructions which enables a change of viewpoint. Fig. 12 shows a close-up of different perspectives created from the same frame captured with our rig. For instance, the parallax between the foreground person and the metallic stool is clearly visible. By fixing points at infinity and animating, we can achieve an effect that corresponds to a sideways head motion. The accompanying video shows a real-time version of this effect displayed on an Oculus head mounted display.

Compositing with CG. For mixing real-world and synthetic content in VR applications, it is straightforward to use the output panorama frames as dynamically changing environment maps for illuminating synthetic objects or refraction mapping (see Fig. 17 for two examples). To insert computer generated content into the output video panoramas, we use the projection derived in Sec. 5.1,

Fig. 13. Stereoscopic panoramas created with our approach, shown as red-cyan anaglyph images. Please see the accompanying video.

Fig. 14. The close-ups show that the panorama created with the depth based approach (top) contains considerably more ghosting artifacts than the panorama created with our flow-based approach (bottom).

which renders objects appropriately distorted in agreement with our panoramic non-linear projection (see Fig. 18).

Comparison to a depth based pipeline. Our approach allows to create omnistereoscopic videos without explicit knowledge of scene geometry. However, if scene geometry was known, performing the view interpolation described in Sec. 4.2 would simply come down to rendering the scene with a common pinhole camera model as in Eq. 3. Therefore at a first glance it may seem more desirable to follow a depth based pipeline that estimates the 3D geometry of the scene and then reconstructs the desired lightfield by rendering novel views. We compare a depth-based result to ours in Fig. 14.

In practice, a depth-based approach suffers from a number of drawbacks: The kind of camera rigs used for creating omnistereoscopic video are not suitable for the popular and commonly available multi-view stereo approaches such as [Fuhrmann et al. 2014; Furukawa and Ponce 2010; Galliani et al. 2015; Zhang et al. 2009] which all did not yield any useable results for our scenario. This may be due to the fact that such methods are rather tailored to scenarios where a large number of cameras observes a part of a scene from different locations as opposed to a relatively small number of cameras observing different parts of a scene from almost the same location.

Thus for a depth based pipeline, we had to resort to a two-view stereo method. We used the OpenCV implementation of a commonly available state-of-the-art stereo algorithm [Hirschmüller 2008]. The depth map for a single camera is estimated in four steps: First, we select the reference camera and its left neighbor, perform a rectification and compute stereo disparities. Second, we do the same for the reference camera and its right neighbor. In the third step, we convert the disparities to depth values using the cameras focal length and baselines, and blend them in the coordinate frame of the reference camera. In order to achieve a reasonable blending, we have to keep track of the overlap regions between camera pairs. In the fourth and last step, we fill in missing information in the depth map by performing inpainting. Fig. 15 visualizes the depth map resulting from these computations for one of the input views.

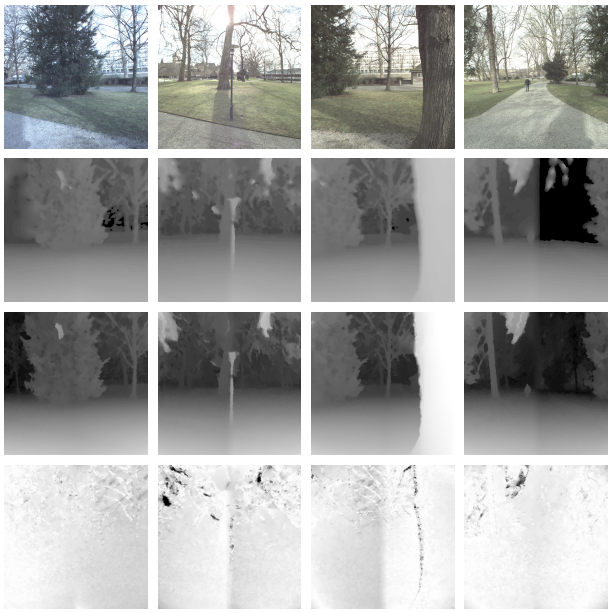

Fig. 15. Row 1: Input images. Row 2: Color coded inverse depth maps. Row 3: Magnitude of optical flow according to Equation 10. Row 4: Deviation of optical flow from epipolar constraints. The maximal distance is 7 pixels and the darker the color the higher the distance. The flexibility to deviate from the epipolar constraint offered by optical flow allows to be more robust and to obtain a higher quality panoramic image (see Fig. 14.)

Besides requiring multiple steps for obtaining a single depthmap, the depth based approach is also much more dependent on accurate calibration information. In the extreme case where no calibration information is used, our flow-based approach is already able to produce photometrically accurate panoramas that only suffer from geometric distortion (as shown in Fig. 11) while a depth based approach is not applicable at all. The depth based approach is sensitive to calibration errors since correspondences are only searched for along epipolar lines. Thus stereo methods cannot deal with imperfections in the calibration and have less flexibility to resolve brittle situations that do not follow epipolar geometry. In contrast, pixel movement described by optical flow can often be sufficient and more stable in practice. The ability to deviate from epipolar constraints allows us to be more robust and remove ghosting artifacts which leads to a visually more pleasing panoramic image (as shown in Fig. 14). Fig. 15 (bottom) illustrates how much the flow correspondences deviate from the epipolar constraints by visualizing our flow-based correspondences decomposed into magnitude along and accross the epipolar line.

Relation to Google Jump. Google Jump [Anderson et al. 2016] and our work have been developed concurrently. Both can be seen as extensions of Megastereo [Richardt et al. 2013] to dynamic scenes. As such, they have to deal with an order of magnitude less input views. This makes the correspondence estimation, which is required for correct view interpolation, more difficult. Both works use flow correspondences instead of depth for view interpolation. While Google Jump uses block matching and the bilateral solver for filtering matches in a second step, we find correspondences by dense optical flow incorporating prealignment.

Besides this, we also derive the omnistereoscopic panorama generation from a view interpolation problem and quantify the deviation to geometric view interpolation that their and also our approach introduce in more detail (see Fig. 7). This gives insight into deviations from geometric view interpolation with respect to the depth of a scene point for cameras with different fields of view, camera arrays with different numbers of cameras, and points with different elevations.

We also describe how to achieve a virtual head motion effect for sideways headmotion in VR and show results in our demo application using an Oculus Rift. Although we suggest using flow correspondences for view interpolation as in Google Jump, we also describe a depth based approach for omnistereoscopic panorama generation and show comparisons between the depth based and the flow-based approach. We also describe and show an example of how to correctly incorporate CG elements into the panorama (Fig. 18).

Comparison to Facebook Surround 360. Fig. 16 compares our method to results obtained with Facebook's recently released Surround 360 software (github.com/facebook/Surround360). We ran the full pipeline of Surround 360 including calibration steps such as the ring rectification and color adjustment. Furthermore, we supplied the intrinsic camera parameters that we also use in our algorithm. Judging by the source code, the ring rectification used in the Surround 360 algorithm jointly estimates a homography for every input camera in order to compensate for small deviations from an ideal camera setup which would be obtained by evenly sampling α in Eq. 3. However, the objective function used in the ring rectification step only penalizes deviations in the y-coordinate of corresponding points after projecting them on a spherical surface. Although the camera rig that we used (see Fig. 9) is already quite close to such an ideal setup, compared to our approach the Surround 360 algorithm produces noticeable artifacts in many regions of the resulting panorama (see the close-ups in Fig. 16).

Limitations. Our approach has some limitations and potential directions for future work. Our study provides insight guidelines for scene acquisition (Sec. 5), for example in terms of minimum scene depth. When the derived minimum depth is violated, ghosting effects appear (Fig. 19).

The admissible range of virtual head motion depends on the radius of the camera array and the amount of camera overlap. To increase the overlap, a solution is to increase the number of cameras, which can become complicated in practice, or to use cameras with a wider field of view. Since a wider field of view corresponds to a loss in spatial resolution, a compromise has to be found in practice.

Creating a smooth head motion animation needs 10 to 20 panoramic image slices. Compared to the usual 2 slices required for common onmnistereo, this is up to an order of magnitude more data which would make it difficult to use for video streaming. However, since there is a lot of redundancy in the data, it is optimally suited for efficient compression. Therefore, investigating compression approaches

Fig. 16. The panorama created with Facebook's Surround 360 algorithm (top) contains visible artifacts that are not present in our panorama (bottom). This is especially visible in the close-ups.

Fig. 17. Application of our results for dynamic environment map for objects with glossy (left) and diffuse (right) material.

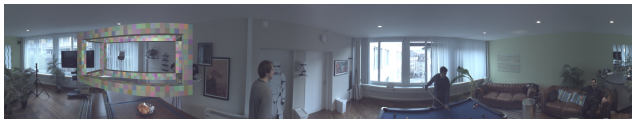

Fig. 18. Example of adding virtual content into a multi-perspective panorama.

for such kind of data or also different representations of this data can be interesting directions for future research. There are also research opportunities for more principled future work, such as the extension of virtual head motion effects beyond sideways translation and handling the singularities at the poles.

7 CONCLUSION

A practically feasible approach for omnistereoscopic video capture and display of the real-world, including support for virtual head motion currently is one of the key hindrances to a more widespread adoption of real-world VR. In this paper, we built on the concept of

Fig. 19. Failure case: ghosting effect of an object too close to the camera array.

omnistereoscopic panoramas and described both, a practical way of capturing and generating the necessary image data from a sparse array of cameras, as well as an analysis of the system parameters and a practical guide to achieve plausible stereoscopic output. By sharing our new insights, we hope that this work facilitates the widespread adoption and creation of real-world VR, given that this is currently such a thriving and exciting area in both academia and industry.

ACKNOWLEDGMENTS

We are very grateful to Henning Zimmer for his tremendous help with the acquisition and his expertise on optical flow. We also thank Max Grosse for his invaluable help with the Oculus Rift. We also would like to thank Maurizio Nitti for creating the Robot scene, Alessia Marra for the experiments with the environment maps, and Jan Wezel for his help with building the camera rigs.

REFERENCES

Agarwala, A., Agrawala, M., Cohen, M., Salesin, D., and Szeliski, R. 2006. Photographing long scenes with multi-viewpoint panoramas. *TOG (SIGGRAPH) 25,* 3, 853–861.

Anderson, R., Gallup, D., Barron, J. T., Kontkanen, J., Snavely, N., Esteban, C. H., Agarwal, S., and Seitz, S. M. 2016. Jump: virtual reality video. *TOG (SIGGRAPH Asia) 35,* 6, 198:1–198:13.

Birklbauer, C. and Bimber, O. 2014. Panorama light-field imaging. *CGF (Eurographics) 33,* 2, 43–52.

Brown, M. and Lowe, D. G. 2007. Automatic panoramic image stitching using invariant features. *IJCV 74,* 1, 59–73.

Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. 2004. High accuracy optical flow estimation based on a theory for warping. In *ECCV*. 25–36.

Chapdelaine-Couture, V. and Roy, S. 2013. The omnipolar camera: A new approach to stereo immersive capture. In *ICCP*. 1–9.

Couture, V., Langer, M. S., and Roy, S. 2011. Panoramic stereo video textures. In *ICCV*. 1251–1258.

Efros, A. A. and Freeman, W. T. 2001. Image quilting for texture synthesis and transfer. In *SIGGRAPH*. 341–346.

Fuhrmann, S., Langguth, F., and Goesele, M. 2014. MVE – a multi-view reconstruction environment. In *Eurographics Workshop on Graphics and Cultural Heritage*.

Furukawa, Y. and Ponce, J. 2010. Accurate, dense, and robust multiview stereopsis. *TPAMI 32,* 8, 1362–1376.

Galliani, S., Lasinger, K., and Schindler, K. 2015. Massively parallel multiview stereopsis by surface normal diffusion. In *ICCV*. 873–881.

Gupta, R. and Hartley, R. I. 1997. Linear pushbroom cameras. *TPAMI 19,* 9, 963–975.

Hartley, R. and Zisserman, A. 2004. *Multiple View Geometry in Computer Vision*. Cambridge University Press.

Hedman, P., Alsisan, S., Szeliski, R., and Kopf, J. 2017. Casual 3D photography. *TOG (SIGGRAPH Asia) 36,* 6, 234:1–234:15.

Hirschmüller, H. 2008. Stereo processing by semiglobal matching and mutual information. *TPAMI 30,* 2, 328–341.

Ishiguro, H., Yamamoto, M., and Tsuji, S. 1992. Omni-directional stereo. *TPAMI 14,* 2, 257–262.

Jia, J. and Tang, C.-K. 2008. Image stitching using structure deformation. *TPAMI 30,* 4, 617–631.

Kang, S. B., Szeliski, R., and Uyttendaele, M. 2004. Seamless stitching using multi-perspective plane sweep. Tech. Rep. MSR-TR-2004-48, Microsoft Research.

Kopf, J., Chen, B., Szeliski, R., and Cohen, M. 2010. Street slide: browsing street level imagery. *TOG (SIGGRAPH) 29,* 4, 96:1–8.

Kopf, J., Uyttendaele, M., Deussen, O., and Cohen, M. F. 2007. Capturing and viewing gigapixel images. *TOG (SIGGRAPH) 26,* 3, 93.

Lee, J., Kim, B., Kim, K., Kim, Y., and Noh, J. 2016. Rich360: optimized spherical representation from structured panoramic camera arrays. *TOG (SIGGRAPH) 35,* 4, 63.

Matzen, K., Cohen, M. F., Evans, B., Kopf, J., and Szeliski, R. 2017. Low-cost 360 stereo photography and video capture. *ACM Trans. Graph. 36,* 4, 148:1–148:12.

Peleg, S. and Ben-Ezra, M. 1999. Stereo panorama with a single camera. In *CVPR*. 1395–1401.

Peleg, S., Ben-Ezra, M., and Pritch, Y. 2001. Omnistereo: Panoramic stereo imaging. *TPAMI 23,* 3, 279–290.

Perazzi, F., Sorkine-Hornung, A., Zimmer, H., Kaufmann, P., Wang, O., Watson, S., and Gross, M. H. 2015. Panoramic video from unstructured camera arrays. *CGF (Eurographics) 34,* 2, 57–68.

Rav-Acha, A., Engel, G., and Peleg, S. 2008. Minimal aspect distortion (MAD) mosaicing of long scenes. *IJCV 78,* 2-3, 187–206.

Richardt, C., Pritch, Y., Zimmer, H., and Sorkine-Hornung, A. 2013. Megastereo: Constructing high-resolution stereo panoramas. In *CVPR*. 1256–1263.

Román, A. and Lensch, H. P. A. 2006. Automatic multiperspective images. In *Eurographics Symposium on Rendering Techniques (EGSR)*. 83–92.

Seitz, S. M. and Kim, J. 2002. The space of all stereo images. *IJCV 48,* 1, 21–38.

Shum, H. and He, L. 1999. Rendering with concentric mosaics. In *SIGGRAPH*. 299–306.

Shum, H. and Kang, S. B. 2000. Review of image-based rendering techniques. In *Visual Communications and Image Processing*. 2–13.

Shum, H., Ng, K. T., and Chan, S. 2005. A virtual reality system using the concentric mosaic: construction, rendering, and data compression. *IEEE Transactions on Multimedia 7,* 1, 85–95.

Shum, H. and Szeliski, R. 2000. Systems and experiment paper: Construction of panoramic image mosaics with global and local alignment. *IJCV 36,* 2, 101–130.

Simon, A., Smith, R. C., and Pawlicki, R. R. 2004. Omnistereo for panoramic virtual environment display systems. In *IEEE VR*. 67–74.

Szeliski, R. 2006. Image alignment and stitching: a tutorial. *Foundations and Trends in Computer Graphics and Vision 2,* 1, 1–104.

Zhang, F. and Liu, F. 2014. Parallax-tolerant image stitching. In *CVPR*. 3262–3269.

Zhang, F. and Liu, F. 2015. Casual stereoscopic panorama stitching. In *CVPR*. 2002–2010.

Zhang, G., Jia, J., Wong, T., and Bao, H. 2009. Consistent depth maps recovery from a video sequence. *TPAMI 31,* 6, 974–988.

Zhang, Z. 2000. A flexible new technique for camera calibration. *TPAMI 22,* 11, 1330–1334.

Zheng, E., Raguram, R., Georgel, P. F., and Frahm, J. 2011. Efficient generation of multi-perspective panoramas. In *International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*. 86–92.

In this appendix, we provide the detailed derivations of some equations presented in the main paper.

A PROJECTION EQUATION

Here, we provide the derivation of the panoramic image formation in Sec. 5.1. When fixing the angle α of the camera along the circle, the projection of Eq. 3 allows to project a world point $\mathbf{X}$ by

$$\begin{pmatrix} x \\ y \end{pmatrix} \cong \mathsf{P}(\alpha)\mathbf{X} \text{ with } \mathsf{P}(\alpha) = \mathsf{K}[\mathsf{R}(\alpha)|\mathbf{t}] \tag{18}$$

where (x, y) is the projection in inhomogenous coordinates. As described in Sec. 4.1 of our paper, the intrinsic calibration matrix is defined as

$$\mathsf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{19}$$

where f_x and f_y denote the focal length in pixels in the horizontal and vertical directions, and (c_x, c_y) the principal point. The pose of each camera is defined as

$$\mathsf{R}(\alpha) = \begin{bmatrix} -\sin(\alpha) & 0 & -\cos(\alpha) \\ 0 & 1 & 0 \\ \cos(\alpha) & 0 & -\sin(\alpha) \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} 0 \\ 0 \\ -r \end{bmatrix} \tag{20}$$

where r is the circle radius. Since a panorama is just a (α, y) slice of L obtained by fixing x, the panoramic camera model can be obtained by fixing x instead of α and solving for (α, y). Given a 3D world point $\mathbf{X} = (X, Y, Z)$, the goal is to compute the camera angle α that will project the 3D world point into the target column x. The x-coordinate of the projection of $\mathbf{X}$ in the image by the projection matrix P is obtained by:

$$\hat{x} = \frac{\mathsf{P}_1(\alpha)[\mathbf{X}, 1]^T}{\mathsf{P}_3(\alpha)[\mathbf{X}, 1]^T} \tag{21}$$

where P_i is the i-th row of P. Constraining $\hat{x} = x$ leads to the system:

$$x \cdot \mathsf{P}_3(\alpha)[\mathbf{X}, 1]^T = \mathsf{P}_1(\alpha)[\mathbf{X}, 1]^T \tag{22}$$

which gives

$$\begin{aligned} & -x \cdot (r - X \cdot \cos(\alpha) + Z \cdot \sin(\alpha)) && \text{(23a)} \\ = \; & X \cdot (c_x \cdot \cos(\alpha) - f_x \cdot \sin(\alpha)) - c_x \cdot r && \text{(23b)} \\ & -Z \cdot (f_x \cdot \cos(\alpha) + c_x \cdot \sin(\alpha)) && \text{(23c)} \end{aligned}$$

This leads to an equation of the form

$$A \sin(\alpha) + B \cos(\alpha) = C \tag{24}$$

with the coefficients

$$\begin{aligned} A &= X \cdot f_x - Z \cdot (x - c_x) && \text{(25a)} \\ B &= Z \cdot f_x + X \cdot (x - c_x) && \text{(25b)} \\ C &= -r \cdot (x - c_x) && \text{(25c)} \end{aligned}$$

Two solutions exist:

$$\begin{aligned} \alpha_1 &= \phi - \gamma + 2\pi k && \text{(26a)} \\ \alpha_2 &= \pi - \phi - \gamma + 2\pi k && \text{(26b)} \end{aligned}$$

where $\phi = \sin^{-1}\left(\frac{C}{D}\right)$, $D = \sqrt{A^2 + B^2}$, $\gamma = \tan^{-1}\left(\frac{B}{A}\right)$, and k is an integer. To obtain α, we set $k = 0$ and simply pick the solution for which $\mathbf{X}$ lies in front of the camera.

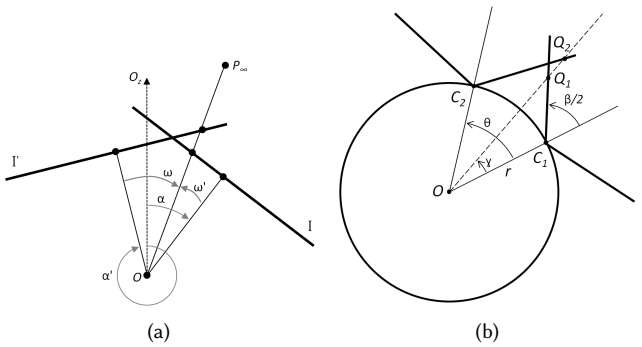

Fig. 20. (a) Illustration for fixing points at infinity. (b) Diagram of the minimum distance. See text for details.

B FIXING PANORAMA ORIENTATION

As discussed in Sec. 5.5, the orientation of the generated panoramas must be consistent for head-mounted displays, and we provided the amount by which the panoramas have to be fixed.

Having panoramas with the same orientation means that points at infinity must be fixed in the generated panoramas, i.e. be at the same location. Let $\mathcal{P}$ and $\mathcal{P}'$ be two panoramas generated from the columns x and x'. In Fig. 20a, P_∞ is a point at infinity and in that case, the acquisition circle radius become negligible, then the camera centers collapse to the circle center O. P_∞ is observed with an angle disparity $\omega = \omega(x)$ and $\omega' = \omega(x')$ in the image planes I and I' of two cameras at angle α and α'. The fact that the point at infinity must be at the same location in the panoramas $\mathcal{P}$ and $\mathcal{P}'$ means that we want $\alpha' = \alpha$. From Fig. 20a, to get the "directions" of α' and α aligned, the direction of α' must be rotated by $\omega' - \omega$. Concretely, by shifting the panorama $\mathcal{P}$ by this angle, then the two panoramas $\mathcal{P}$ and $\mathcal{P}'$ have a consistent orientation.

C MINIMUM VISIBLE DEPTH

In Sec. 5.3, we presented the equation of the minimum visible depth. Fig. 20b illustrates the minimum distance observed by two cameras C_1 and C_2 with field of view β, on a circle of radius r centered at O. The angle between the optical axis of the cameras is noted θ. Given an angle γ (see Fig. 20b), the minimum distance observed by C_1 is $d_1 = \|OQ_1\|$, and by C_2 is $d_2 = \|OQ_2\|$. These distances can be computed by

$$\begin{aligned} d_1 &= r\frac{\sin(\pi - \beta/2)}{\sin(\beta/2 - \gamma)} && \text{(27a)} \\ d_2 &= r\frac{\sin(\pi - \beta/2)}{\sin(\beta/2 - (\theta - \gamma))} && \text{(27b)} \end{aligned}$$

Thus the minimum distance observed by the two cameras is $\max(d_1, d_2)$, which is obtained when $\gamma = \theta$ in practical settings. The minimum distance observed by the two cameras is therefore:

$$d = r\frac{\sin(\pi - \beta/2)}{\sin(\beta/2 - \theta)}. \tag{28}$$

Received XXX; revised XXX; final version XXX; accepted XXX