A Low-Cost Portable Polycamera for Stereoscopic 360° Imaging

Hong-Shiang Lin, Yung-Yu Chuang, Ming Ouhyoung

***Abstract*—This paper proposes a low-cost and portable polycamera system and accompanying methods for capturing and synthesizing stereoscopic 360° panoramas. The polycamera consists of only four cameras with fisheye lenses. Synthesizing panoramas from only four views is challenging because the cameras view very differently and the captured images have significant distortions and color degradation including vignetting, contrast loss and blurriness. For coping with these challenges, this paper proposes methods for rectifying the polyview images, estimating depth of the scene and synthesizing stereoscopic panoramas. The proposed camera is compact in size, light in weight and inexpensive in cost. The proposed methods allow synthesizing visually pleasing stereoscopic 360° panoramas using the images captured with the proposed polycamera. We have built a prototype polycamera and tested it on a set of scenes with different characteristics on depth ranges and depth variations. The experiments show that the proposed camera and methods are effective for generating stereoscopic 360° panoramas that can be viewed with popular virtual reality displays.**

***Index Terms*—Stereoscopic 360° cameras, Omnistereo panoramas, Polycameras.**

I. Introduction

Virtual reality (VR) enables users to navigate through an artificial world and offers novel ways that users can interact with others and the digital world. VR has become very popular recently. Quite a few head-mounted displays are available on the market at affordable prices [1]–[3]. With these devices, VR is no longer a privilege for scientists and developers to play with, but has become accessible to general consumers for daily use. In addition to stereoscopic 3D, an important characteristic of VR is to allow users to look around [4]. For synthetic scenes, it is generally not a problem to synthesize the scene from different viewpoints. However, for real scenes, it would require omnidirectional capture for supporting looking around. It can be achieved via stitching softwares [5], [6], multi-view stitching systems [7] or omnidirectional cameras [8]. Although there are more and more 360° cameras on the market, most of them can only capture a single panorama. Thus, when viewing their captured images with VR displays, although allowing looking around, there is no stereoscopic 3D since both eyes see the same image.

There were a few attempts for making stereoscopic 360° cameras. Some researchers proposed to use a rotating camera for constructing an omnistereo panorama [9]–[11]. Although effective on producing stereoscopic 360° panoramas, the system with a rotating camera can only capture static scenes since it takes time to rotate the camera in a full circle. To overcome the limitation, based on Peleg et al.'s design [10], several later researches employed multiple synchronized optical systems [12]–[14]. These systems are however often bulky and costly. For example, *Google Jump* [14], [15] consists of 16 GoPro cameras on a circle with the diameter 280 mm and weights approximately 6.5 kg, making it cumbersome to carry and operate. In addition, it costs roughly $\$10,000$. Facebook Surround 360 [16] has a similar design with 17 cameras and the estimated cost of $\$30,000$. Although offering high-quality images, the relative high cost and low portability make them more suitable for professionals but not the general consumers.

This paper proposes the design of a stereoscopic 360° camera with lower cost and better portability. To reduce the cost and enhance the portability, we aim to use much fewer cameras, resulting in much fewer views, as compared to previous designs such as Google Jump. To encode stereoscopy on fully spherical field of view (FOV), the input views together should span over at least 2 times $360° \times 180°$ field of view. A possible minimal configuration is to use three cameras with ultra wide-angle fisheye lenses, each with $240°$ horizontal FOV. However, ultra-wide angle lenses are more expensive than fisheye lenses with $180° \times 180°$ to $190° \times 190°$ FOV and often produce even worse image quality. Therefore, the proposed system consists of four fisheye cameras. The fisheye cameras are placed on a circle with the diameter 100 mm and the viewing directions of neighboring cameras are roughly orthogonal to each other. Each fisheye camera captures the $190° \times 190°$ FOV. A camera shares almost one half of its visual field with each of its neighboring camera. The compact size and the lightweight design make the camera more portable. The design is similar to the *polycamera* [17] which was designed to capture monocular panoramas by packing fisheye cameras with much smaller FOV. Thus, we use the same name, polycamera, to refer to the proposed camera system. Figure 1(a) shows a prototype of the proposed polycamera and an example of the captured polyview image consisting of four fisheye images. With the captured images, this paper proposes methods for synthesizing two 360° panoramas respectively for the left and right eyes, shown in Figure 1(b). Together, they form a stereoscopic 360° panorama; and when viewing with VR displays, users can view the captured scene from any direction with vivid depth perception (Figure 1(c)).

Although the use of few views can really lead to a polycamera system which is easy to carry and inexpensive, it also presents great challenges for omnistereo panorama synthesis. Previous methods often require dense views for synthesizing high-quality panoramas. For example, the rotating slit camera [10] would require a large number of views to reduce cross-view distortions and visible seams caused by the limited angular resolution [18]. The flow-based view

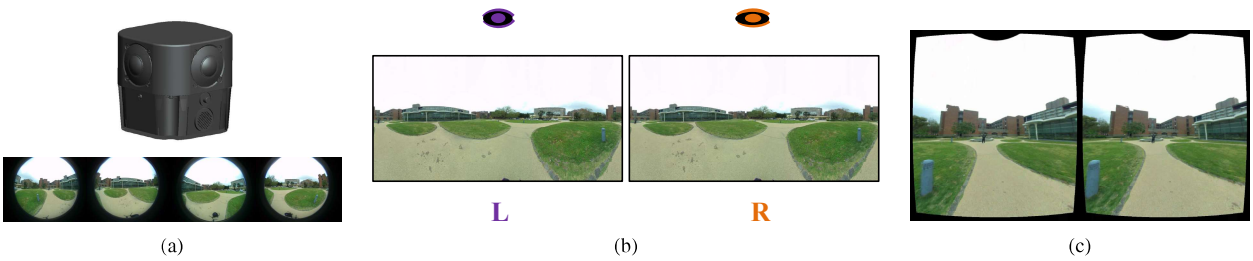


Fig. 1. Overview of the proposed camera, the synthesized panoramas and the VR viewing application. (a) A prototype of the proposed polycamera and its captured fisheye images (for the example *Library*). (b) The two 360° panoramas respectively for the left and right eyes, synthesized from the captured fisheye images using the proposed stereoscopic panorama synthesis method. (c) A screenshot of the mobile phone for stereoscopic viewing when using the Google VR Cardboard.

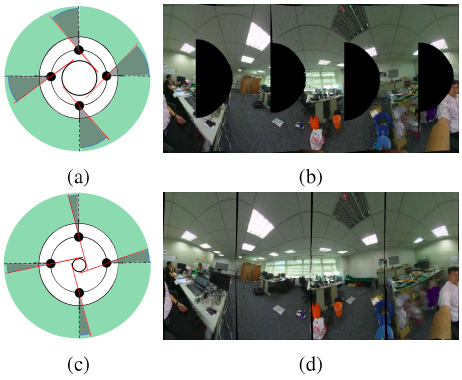


Fig. 2. The problem with holes for flow-based interpolation on sparse views. The black dots and arrows denote the positions and viewing directions of cameras. The black circles from inner to outer are the image circle, the circle of the camera placement and the viewing circle. The red lines denote the sampled image strips on input views for the right-eye panorama composition, and the black dotted lines denote the boundaries of overlapped FOVs between neighboring views. Correspondences can be computed within the regions with view overlaps (the green regions). For non-overlap regions (gray regions), there is no correspondence and view interpolation cannot be taken. When setting the radius of the viewing circle at 45mm as our setting (the top view (a)), there are significant holes in the synthesized panorama (b) with flow-based interpolation. By decreasing the radius to 1.4mm (c), the holes can be reduced in the panorama (d), but the 3D effects are also significantly reduced with less disparity.

interpolation method used by Jump [14] and Surround 360 [16] also requires dense views to work. If views are sparse, flow-based interpolation could suffer from the problem with holes. Figure 2 illustrates such a problem. The holes occur within non-overlapped FOVs because there is no correspondence and no interpolation can be performed. It is possible to reduce the holes by decreasing the separation between sampled image strips and the central strips of input views. However, doing so leads to less disparity values on the omnistereo panorama (with a smaller radius for the viewing circle), thus reducing 3D effects.

To use very few views for view synthesis, the key idea is to recover the depth information encoded in the overlapped FOV between cameras and use depth-image-based rendering for view synthesis. For having sufficient view overlaps with only four cameras, a much wider FOV is required for each camera, leading to the use of the fisheye lens. Accompanied with the advantage of providing view overlaps, one has to deal with the problems of significant image distortions and color degradation caused by the fisheye lens. In addition, our viewing directions are outward and nearly orthogonal to each other. Such viewing configuration results in large non-overlapped visual fields between neighboring views, thus increasing the matching ambiguity of stereo correspondence estimation. To address the above challenges, this paper proposes methods for polyview image rectification, panoramic depth estimation and stereoscopic panoramic view synthesis.

- Polyview image rectification. The first step is to undistort and align images so that image disparity values can be inferred robustly along horizontal scanlines. As the first step, the paper proposes a robust and fully automatic polycamera calibration method for camera pose recovery and lens distortion estimation. Next, the paper proposes a compact spherical stereo view transformation to rectify fisheye images into a set of inverse-equirectangular image pairs. The resulting view representation effectively records stereo information of fully spherical field of view and removes non-overlapped regions for each image pair.
- Panoramic depth estimation. The second step is to estimate scene depth from a set of rectified inverse-equirectangular image pairs. The images exhibit significant color degradation such as blurriness and vignetting with large fisheye projection angles. Instead of applying separate color transform steps which depend heavily on quality of image alignment, the paper proposes a trinocular matching cost blending method which aggregates color information from all views with an adaptive weighting scheme on fisheye projection angles. The algorithm greatly reduces depth discrepancy across stereo visual fields. Furthermore, the paper presents an angular disparity transformation across different stereo pairs, yielding an effective correspondence reprojection.
- Stereoscopic panorama synthesis. This paper presents dedicated depth-image-based rendering methods for omnistereo panorama synthesis. We derive formulations for efficient forward and inverse mapping by exploring the omnistereo projection model. In addition, the inverse mapping incorporates depth consistency to resolve colliding candidates, greatly reducing artifacts on object boundaries.

The paper is organized as follows. In Section II, we briefly review related work. The next three sections describe methods for polyview image rectification (Section III), panoramic depth

	Camera Configuration	Panorama Resolution	Price	Portablity	Stitching Artifacts	Effective Disparity
Google Jump [14]	16 cameras on a ring	8K by 4K	High	Low	No ghosting but the method requires dense views to work	Effective at all directions
Surround360 [16]	17 cameras on a ring	8K by 4K	High	Low	No ghosting but the method requires dense views to work	Effective at all directions
NOKIA OZO [36]	8 cameras on a spherical rig	4K by 2K	High	Low	No ghosting but the method requires dense views to work	Effective at all directions
Fraunhofer HHI OmniCam [21]	10 mirrors with 20 micro cameras	10K by 2K	High	Low	No ghosting but the method is limited to mirror-based systems	Effective at all directions
Samsung Gear [37]	2 cameras back to each other	4K by 2K	Low	High	Expose Ghosting	No disparity (monocular)
Low-cost 360 photography [23]	4 cameras on a stereo rig	6K by 3K	Low	High	No ghosting	No disparity along the rig baseline
The proposed polycamera	4 cameras on a ring	6K by 3K	Low	High	No ghosting	Effective at all directions

TABLE I
COMPARISONS OF REPRESENTATIVE 360 DEVICES/METHODS ON THEIR RESOLUTIONS, PRICES, PORTABILITY, STITCHING ARTIFACTS, AND THE VIEWING RANGES OF EFFECTIVE DISPARITY.

reconstruction (Section IV) and stereoscopic panorama synthesis (Section V) respectively. Section VI provides evaluation on the proposed methods and presents several synthesized stereoscopic 360° panoramas. Finally, Section VII concludes the paper and suggests future work.

II. RELATED WORK

Omnistereo cameras. Peleg et al. proposed a projection model and an early camera design for capturing omnistereo panoramas [10], [11]. Their rotating slit camera system aggregates two-sided strips of every captured image to construct a pair of cylindrical panoramas. The stitching method however could suffer from artifacts of visible seams and vertical parallax due to the imperfection of camera setting in practice and the limited angular resolution. Richardt et al. addressed these issues by using structure from motion to recover camera poses and performing cylindrical rectification to remove the image distortion and vertical parallax [19]. Their system further computes *net flow field* to guide the image blending. A major drawback of the rotating camera system is that it can only capture static scenes. Aggarwal et al. presented *Coffee-filter*, a customed mirror system which can capture panoramic stereo videos without rotating optics [20]. However, it can only synthesize images of a medium vertical resolution.

To handle dynamic scenes in general, several people have proposed synchronized camera systems. Tanaka and Tachi realized Peleg et al.'s omnistereo model [10] with a rotating optic system composed of prism sheets, polarizing films, and a hyperbological mirror [12]. It constructs a complete panorama with five frames with a high-speed shutter. The Fraunhofer Institute proposed a mirror-based multi-camera system which consists of ten 36° mirrors and twenty micro cameras (two cameras for each mirror) [21]. It allows parallax-free stitching but the vertical field of view is limited. Amini et al. aggregated ten stereo cameras around a horizontal circle and stitched images by solving homographies among cameras [13]. This stitching method could introduce inconsistent distortions between the left and right panoramas. *Google Jump* [14] synchronizes 16 GoPro cameras and uses a dedicated optical flow method for efficient view interpolation, generating visually pleasing results. The above systems use many cameras. Thus, they are costly and often cubersome to carry and set up. Chapdelaine-Couture et al. proposed to use few cameras using an omnipolar system with wide-angle lens [22]. It can capture omnistereo with a minimal set of three cameras. However, the horizontal disparity degenerates around the stitching boundary. A very recent research aimed to propose a low-cost device which consists of two 360° cameras on a stereo rig [23]. The camera however cannot simulate proper stereo at all directions. Later, Robert et al. presented *Vortex* [24], a rotated stereo capture with a high-speed motor which can natively handle challenging phenomena such as refraction and reflection. However, the resulting images tend to be dark due to high-speed spins and shorter exposures of the cameras. Table I compares the proposed polycamera with other representative 360 devices/methods.

Omnidirectional camera calibration. There were calibration systems [25]–[27] for panoramas generated from rotational line scanning cameras such as *SpheroCam* [28]. Schneider et al. adopted a general rotation model considering imperfect camera rotation such as eccentricity of the projection center, non-parallelism CCD lines, and deviation from the planar move [25]. The deployment consists of hundreds of targets in a calibration room. Parian et al. formulated a 3D straight line constraint on a panoramic image with multiple viewpoints to reduce the required number of feature points for calibration [26]. Guan et al. simplified the calibration environment with a calibration box, and formulated a multi-plane projection onto a sphere [27]. The system adopts a single viewpoint projection model and only estimates vertical field of view of the spherical image.

Omnidirectional depth estimation. There were quite a few omnidirectional systems aiming to reconstruct scene depth within a single shot [29]–[32]. These systems aim for real-time depth sensing via vertical disparity computation. Nayar proposed to use two specular balls to record light rays and a perspective camera to capture the specular reflection [29], while Southwell et al. used double lobed mirrors [30]. Both

systems have multiple centers of projection, leading to more complicated calibration. Therefore, Cluckman et al. used parabola mirrors with a single center of projection to simplify the calibration [32]. Nene used two mirrors and a single camera to reduce the manufacturing cost [31]. The above devices cannot generate high-resolution depth maps due to the limited image resolution.

There were also researches aiming to acquire omnidirectional depth by matching cameras at multiple viewpoints. Arican et al. formulated a global optimization framework and used graph cut to match single-viewpoint panoramas captured at two different viewpoints [33]. Lee et al. proposed a multi-resolution approach to refine depths in textureless regions [34]. For large-scale scene reconstruction, Schönbein et al. used a stereo omnidirectional camera rig which is able to acquire a complete 360° depth map from two consecutive frames [35]. The proposed system extracts plane candidates from virtual 360° disparity maps and refines depth maps by the global depth plane assignment. All the above systems cannot compute depth for dynamic scenes and require more than two viewpoints for acquiring effective disparity value at every viewing direction.

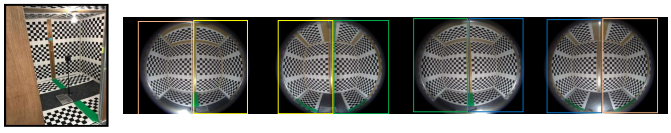
III. Polyview Image Rectification

The first step of our system is to rectify the captured fisheye images (e.g., the top row of Figure 7) into a compact stereo view representation (e.g., the middle row of Figure 7). For the rectification, we first relate cameras by calibration, and then use the recovered camera parameters for view transformation. The proposed calibration procedure includes a simple calibration depolyment and a robust camera parameter estimation method, where the camera parameters are formulated based on our proposed polycamera projection model. The proposed view transformation includes view overlap computation and compact spherical image rectification methods, where the spherical image rectification compactly undistorts input fisheye images into a set of rectified stereo image pairs on the inverse-equirectangular space.

A. Polycamera calibration

Calibration deployment. For our multi-camera wide-angle setting, it would be tedious and time-consuming for moving a chessboard to cover a full 360° × 180° field of view. To ease the calibration process, we design a six-sided calibration cube covered by multiple chessboards (Figure 3(a)). The deployment ensures that each camera captures sufficient uniformly distributed features. With the calibration cube, the polycamera can be calibrated with a single shot[1]. The dimension of the calibration cube is 190 cm × 190 cm × 190 cm, and each chessboard consists of 12 × 6 grids of the dimension 7.5 cm × 7.5 cm. Each of the four horizontally oriented faces is covered with six chessboards, while each of the two vertically oriented faces is only covered with four chessboards. For calibration, the polycamera is put at the center of the calibration cube and each camera is oriented towards the center of each of the four horizontally orientated faces. With this deployment, each fisheye camera roughly captures sixteen chessboards and each pair of neighboring cameras capture roughly eight chessboards in common, as shown in Figure 3.

[1]Although it is possible to move the cameras for capturing more images, experiments show that a single shot is enough for image rectification.



(a) (b)

Fig. 3. Calibration deployment and the captured images. (a) The calibration cube. (b) The four images captured by fisheye cameras in a single shot. Each chessboard is captured by two cameras. The shared views are outlined with the same color.

Polycamera projection estimation First, we describe the polycamera projection model. We use $\mathbf{V}_i$ to denote the i-th fisheye camera in order, where $i = 0..n$, n is the number of cameras and $n = 4$ in our polycamera. Let $\mathbf{P}_i$ denote the projection function for $\mathbf{V}_i$, consisting of the intrinsic projection of each camera and relative poses between cameras. $\mathbf{R}_{i,i+1}$ and $\mathbf{T}_{i,i+1}$ denote the relative rotation and translation between two consecutive cameras $\mathbf{V}_i$ and $\mathbf{V}_{i+1}$ (the addition for the camera index is actually modulo n because $\mathbf{V}_{n-1}$ and $\mathbf{V}_0$ are next to each other). $\mathbf{K}_i$ is the intrinsic projection for $\mathbf{V}_i$ and it incorporates the polynomial model [38] for modeling the wide-angle fisheye lens distortion. Since the fisheye cameras in our polycamera are configured on a loop trajectory, the relative pose between the first and the last cameras, $\mathbf{R}_{n-1,0}$ and $\mathbf{T}_{n-1,0}$, can be further formulated as:

$$\mathbf{R}_{n-1,0} = \prod_{i=0}^{n-2} \mathbf{R}_{i,i+1}^T, \mathbf{T}_{n-1,0} = -\sum_{i=0}^{n-2} (\prod_{j=0}^{i} \mathbf{R}_{j,j+1})^T \mathbf{T}_{i,i+1} \tag{1}$$

The polycamera projection parameters can be estimated by minimizing the reprojection error of chessboard corners. We relate all pairwise transformations with the loop trajectory formulation (Equation 1) and jointly refine all projection parameters $\mathbf{P}_i$ for all views with the Gauss-Newton algorithm[2].

B. View transformation

View overlap determination. The field of views of fisheye lenses and the camera placement determines the stereo viewing coverage of the proposed polycamera. Figure 4 shows the top view of the viewing configuration. Ideally, if all cameras are placed correctly with 180° FOV, each point can be viewed by two cameras. However, the camera placement cannot be perfect. Figure 4 illustrates viewing deviation that could occur. In this case, there could be the "blind matching area" which can only be seen by one camera. Fortunately, fisheye lens usually have around 200° FOV. The problem of imperfect

[2]For incorporating the loop constraint into the bundle optimization, the derivatives of the projection to $\mathbf{R}_{n-1,0}$ and $\mathbf{T}_{n-1,0}$ can be expressed as the combinations of derivatives of the projection to other rotations and translations by using the chain rule.

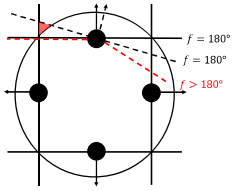
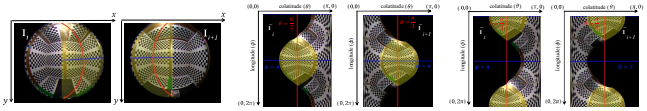


Fig. 4. The top view of the viewing configuration. The solid lines represent 180° horizontal FOV per camera. The dotted arrows and lines represent viewing directions with 180° FOV deviated from the ideal viewing directions represented by solid arrows with 180° FOV. The red region denotes a "blind matching area" resulting from the viewing deviation, which can be remedied by using a lens with a wider FOV denoted by red dotted lines.



(a) (b) (c)

Fig. 5. View overlaps of $\mathbf{V}_0$ and $\mathbf{V}_1$ (denoted by the yellow color) using $190° \times 190°$ FOV for each camera. All red curves (lines) and blue curves (lines) denote colatitudes $\theta = 90°$ and longitudes $\phi = 180°$ on the rectified viewing spheres. (a) The fisheye view overlaps. (b)(c) Spherical rectification results for $\mathbf{V}_0$ and $\mathbf{V}_1$. Yellow regions denote the transformed view overlaps on inverse-equirectangular images. (b) The results of transformed view overlaps. (c) Undesirable split view overlaps. This figure shows the transformation results by reversing the new viewing direction described in the step #2 of the rectification process.

camera placement can be remedied by using larger FOV (red dotted lines in Figure 4). Given a specified FOV for each fisheye image, we compute view overlaps on the fisheye space. For each pixel, we generate the corresponding viewing ray using the recovered intrinsic projection parameters. Then we compute the angle between the viewing ray and the viewing direction of the neighboring view. If the angle is smaller than the largest projection angle, the pixel is marked as an overlapped pixel. Figure 5(a) shows the computed view overlaps for the proposed polycamera.

Compact spherical rectification. Given the calibrated camera parameters and computed fisheye view overlaps, the four fisheye images $\mathbf{I}_i$ can be transformed into a set of rectified images for stereo matching. Since traditional perspective rectification results in severe stretch effects on large projection angles, we choose to rectify the fisheye images on the inverse-equirectangular space rather than the perspective space. This paper proposes a dedicated application of the spherical rectification theory [39] to construct a compact stereo view representation for the proposed polycamera.

The basic idea of spherical rectification is to rotate viewing spheres so that north poles are aligned with the epipoles and a 3D point projects onto the same longitude on the rotated viewing spheres [39]. Our goal is to compute rotation matrices $\tilde{\mathbf{R}}_i$ such that the computed view overlaps are transformed into compact image regions. Figure 6 illustrates the rectification process for $\mathbf{V}_0$ and $\mathbf{V}_1$. First, we build an initial spherical coordinate system for each view. The up vector $\mathbf{u} = [0 \quad -1 \quad 0]^T$ is aligned to the north pole, and the viewing direction $\mathbf{v} = [0 \quad 0 \quad 1]^T$ is towards the greenwich (longitude = 0°). Then $\tilde{\mathbf{R}}_0$ and $\tilde{\mathbf{R}}_1$ are computed by the following procedure where we construct $\tilde{\mathbf{R}}_0$ first and then compute $\tilde{\mathbf{R}}_1$ accordingly:

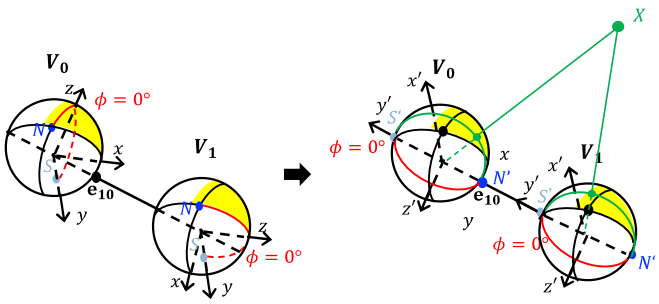


Fig. 6. A 3D view of spherical rectification for the viewing spheres $\mathbf{V}_0$ and $\mathbf{V}_1$ of the proposed camera. Viewing directions and up vectors are not shown here for brevity. After spherical rectification, a 3D point X projects on the same longitudes (green curves) of the viewing spheres. In addition, the view overlaps denoted by yellow colors are located on the regions close to $\phi = 180°$, resulting in a compact transform to the inverse-equirectangular space.

1) Assign the new up vector $\mathbf{u}'$ of $\mathbf{V}_0$ to be aligned with the epipole $\mathbf{e}_{10}$ (i.e., the projection of the viewpoint of $\mathbf{V}_1$ on the viewing sphere of $\mathbf{V}_0$). For achieving this, the second column vector $\mathbf{r_2}$ of $\tilde{\mathbf{R}}_0$ is assigned as $\mathbf{u}'$.
2) Any vector orthogonal to the new up vector can be a candidate of the new viewing direction $\mathbf{v}'$. However, the new viewing direction regards the new greenwich projection. To prevent splitting view overlaps on the rectified images, the new greenwich can be projected to the back of input views. Therefore, we set the new viewing direction $\mathbf{v}' = (\mathbf{v} \cdot \mathbf{e}_{10})\mathbf{e}_{10} - \mathbf{v}$ For achieving this, the third column vector $\mathbf{r_3}$ of $\tilde{\mathbf{R}}_0$ is assigned as $\mathbf{v}'$.
3) Finish the construction of $\tilde{\mathbf{R}}_0$ by computing its first column vector $\mathbf{r_1}$ as $\mathbf{r_2} \times \mathbf{r_3}$.
4) Compute $\tilde{\mathbf{R}}_1 = \mathbf{R}_{0,1}\tilde{\mathbf{R}}_0$ where $\mathbf{R}_{0,1}$ is the extrinsic parameter from calibration.

After rotations, corresponding pixels are located at the same height of inverse-equirectangular images and view overlaps are transformed into compact regions, as shown in Figure 5(b). Finally, we apply the spherical rectification to all image pairs and fit the view overlaps into bounding boxes to remove non-overlapped regions. Figure 7 demonstrates the rectification result for a real scene using the proposed method, where $\tilde{\mathbf{I}}_i^j$ denotes the rectified image of $\mathbf{V}_i$ with respect to the shared view with $\mathbf{V}_j$. The compact stereo view representation retains fully spherical stereo coverage while largely reducing disparity search ranges of the following stereo matching.

IV. Panoramic Depth Reconstruction

To handle the challenge of view synthesis from very sparse views, we leverage the depth information encoded in the overlap of views. This section describes our method for depth estimation. One thing to note is that, for our application, it is not necessary to have accurate depth information everywhere. Since our goal is view synthesis not 3D modeling, the recovered depth can be erroneous at places as long as the synthesized view looks visually plausible. We first outline the requirements of the depth estimation for offering reasonable depth perception in our application.

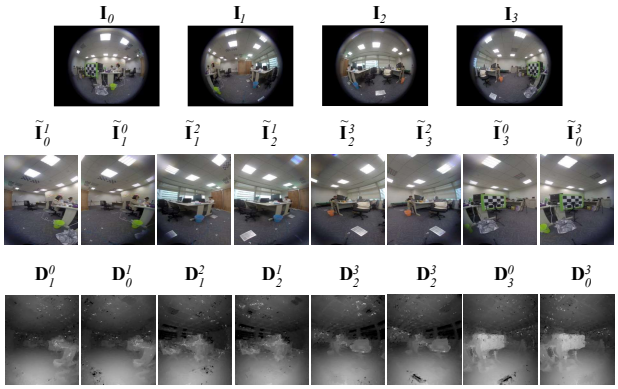


Fig. 7. A polyview depth estimation result for a real scene. The top row shows the four input images captured by fisheye cameras. The middle row shows the compact stereo view representation generated by the proposed compact spherical rectification. The bottom row shows the disparity maps of rectified images by the trinocular matching method.

1) *Depth discontinuity alignment.* Occlusion is arguably the most effective cue for depth perception. We rely heavily on occlusion to sense the depth relationship among objects in a scene. The depth discontinuity should be aligned well with object boundary to enhance depth perception.
2) *Depth smoothness.* We are not sensitive to depth within textureless regions. Depth smoothness is often more crucial than accuracy in these regions.
3) *Cross-view depth consistency.* A viewing space of interest may cross visual fields of different sampled views. It is important to preserve depth continuity across views even if the views are processed separately. This is a particular issue that needs to be addressed in our system.

The proposed method incorporates a trinocular spherical stereo matching framework to preserve cross-view depth consistency (requirement #3), and an edge-aware filtering to smooth out the potentially erroneous initial depth estimation to ensure depth discontinuity alignment (requirement #1) and depth smoothness (requirement #2).

A. Trinocular stereo matching

Given a rectified image $\tilde{\mathbf{I}}_i^j$, the goal of depth reconstruction is to estimate the corresponding disparity map $\mathbf{D}_i^j$, where $\mathbf{V}_i$ is the reference view and $\mathbf{V}_j$ is one of its neighboring views ($j = i-1$ or $j = i+1$). For a hypothesized disparity d for a pixel p on $\tilde{\mathbf{I}}_i^j$, its matching point m_p can be found by $p + d$ (the addition is actually with $(d, 0)$. For brevity, we use $p + d$ for the addition of p and $(d, 0)$.). As introduced in Section III, all rectified images contain a fully stereo viewing coverage outside a viewing sphere with a specified minimum depth value. The specified depth range can be converted into the disparity search range for each image pair. Since d is within the disparity range, we can ensure that p must be visible in at least two views. That is, p with a given d must have a correspondence in $\mathbf{V}_{i-1}$ or $\mathbf{V}_{i+1}$ with the proposed polycamera. Based on this prior we aggregate three views to estimate $\mathbf{D}_i^j$.

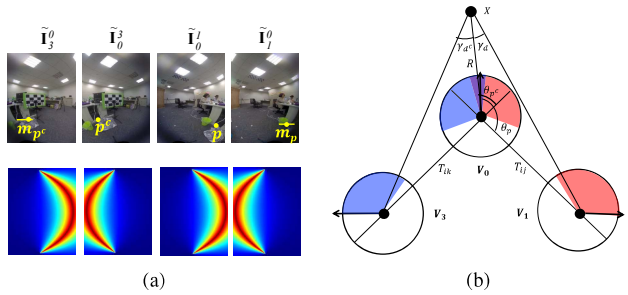


Fig. 8. Cross-view correspondence between the rectified images. (a) Top row: Rectified images related to three views $\mathbf{V}_0$, $\mathbf{V}_1$, and $\mathbf{V}_2$. m_p and m_{p^c} are corresponding points of p (and p^c) given the disparity values d and d^c, respectively. Bottom row: Corresponding pixel confidence maps for the four rectified images. (b) The top view for the 3D triangulation on viewing spheres of $\mathbf{V}_0$, $\mathbf{V}_1$, and $\mathbf{V}_3$. $\tilde{\mathbf{I}}_0^1$ and $\tilde{\mathbf{I}}_1^0$ correspond the view overlap denoted by red regions; $\tilde{\mathbf{I}}_0^3$ and $\tilde{\mathbf{I}}_3^0$ correspond the view overlap denoted by blue regions. The 3D point X in this case is trinocularly visible.

The use of trinocular matching would ensure cross-view depth consistency. It is especially beneficial for the pixels around the border of the cropped view, which is potentially invisible in the neighboring view if binocular matching is used. These pixels also exhibit severe color degradation due to the fisheye lens as shown in Figure 8. Fortunately, they could retain the color better in the other view.

Cross-view disparity transformation. The paper presents a cross-view spherical disparity transformation formulation to transform the correspondence $(p, p+d)$ of $\mathbf{V}_i$ and $\mathbf{V}_j$ ($\tilde{\mathbf{I}}_i^j$ and $\tilde{\mathbf{I}}_j^i$) to (p^c, p^c+d^c) of $\mathbf{V}_i$ and $\mathbf{V}_k$ ($\tilde{\mathbf{I}}_i^k$ and $\tilde{\mathbf{I}}_k^i$), where $\mathbf{V}_k$ is the other neighboring view of $\mathbf{V}_i$. Note that p and p^c represent the same point in $\tilde{\mathbf{I}}_i^j$ and $\tilde{\mathbf{I}}_i^k$ respectively. Let θ_p and θ_{p^c} denote the co-latitudes of p and p^c; γ_d and γ_{d^c} denote the angular disparity values of d and d^c. γ_{d^c} and γ_d are related using the following triangulation formula:

$$R = T_{ij}\frac{\sin(\theta_p + \gamma_d)}{\sin(\gamma_d)} = T_{ik}\frac{\sin(\theta_{p^c} + \gamma_{d^c})}{\sin(\gamma_{d^c})}, \tag{2}$$

where R is the distance from the triangulated 3D point to the camera center of $\mathbf{V}_i$; T_{ij} is the baseline between $\mathbf{V}_i$ and $\mathbf{V}_j$; and T_{ik} is the baseline between $\mathbf{V}_i$ and $\mathbf{V}_k$. Figure 8 demonstrates such relationship by taking $\mathbf{V}_0$, $\mathbf{V}_1$, and $\mathbf{V}_3$ as an example. Based on Equation 2, the disparity can be effectively transformed without the expensive computation of 3D re-projection:

$$\gamma_{d^c} = \tan^{-1}\frac{\sin(\theta_{p^c})}{\frac{T_{ij}\sin(\theta_p+\gamma_d)}{T_{ik}\sin(\gamma_d)} - \cos(\theta_{p^c})}. \tag{3}$$

Matching cost blending. To integrate the information from the three views together, for the given pixel p on $\tilde{\mathbf{I}}_i^j$, we define the cost function $C_i^j(p, d)$ for the hypothesized disparity d as

$$\begin{aligned} C_i^j(p,d) = \quad & v_j(2-v_k)w(p,d)\rho_{i,j}(p,d) \\ & +v_k(2-v_j)w(p^c,d^c)\rho_{i,k}(p^c,d^c). \end{aligned} \tag{4}$$

We use $\rho_{i,j}(p, d)$ to measure the color discrepancy for the hypothesized disparity d,

$$\rho_{i,j}(p,d) = \Delta(\tilde{\mathbf{I}}_i^j(p), \tilde{\mathbf{I}}_j^i(p+d)), \tag{5}$$

where Δ is a function using lighting invariant measures, such as census transform [40] and Sobel transform. $\rho_{i,k}$ is defined similarly. Equation 4 uses v_j and v_k to incorporate the visibilities on $\mathbf{V}_j$ and $\mathbf{V}_k$, respectively. There are two possible visibility conditions for the proposed polycamera:

- case 1: p is visible only in one of the images $\tilde{\mathbf{I}}_j^i$ and $\tilde{\mathbf{I}}_k^i$. In this case, either $(v_j = 1, v_k = 0)$ or $(v_j = 0, v_k = 1)$ holds. Equation 4 becomes either $2w(p,d)\rho_{i,j}(p,d)$ or $2w(p^c,d^c)\rho_{i,k}(p^c,d^c)$. Note that the constant 2 is for balancing the power of this case and case 2.
- case 2: p is visible in both images $\tilde{\mathbf{I}}_j^i$ and $\tilde{\mathbf{I}}_k^i$. In this case, $v_j = 1$ and $v_k = 1$; and Equation 4 reduces to $w(p,d)\rho_{i,j}(p,d) + w(p^c,d^c)\rho_{i,k}(p^c,d^c)$.

We now discuss the weighting function w in Equation 4. It considers both pixel visibility and color confidence, and is defined as follows

$$\begin{aligned} w(p,d) &= \frac{v_i f(p,d)}{v_i f(p,d) + v_k f(p^c,d^c)}, \\ w(p^c,d^c) &= \frac{v_k f(p^c,d^c)}{v_i f(p,d) + v_k f(p^c,d^c)}, \end{aligned} \tag{6}$$

where $f(p,d)$ gives the confidence of the pixel p with the disparity d. Due to vignetting and blurriness of the large fisheye projection angle, we trust the measurement of the pixels closer to the center of the source image more than the ones near the boundary. Thus, for a pixel p, we define its confidence as $e^{-\alpha_p}$ so that it is inversely proportional to its incident angle α_p in its source viewing sphere. Figure 8(a) shows example confidence maps. For a pair of match points, p and $m_p = p + d$, we define the confidence as the smaller one of their confidence values,

$$f(p,d) = \min(e^{-\alpha_p}, e^{-\alpha_{m_p}}), \tag{7}$$

since we want both pixels are reliable. $f(p^c,d^c)$ is defined similarly. Note that, from Equation 6, the weight equals 1 when there is only one visible view (case 1); and the weight is proportional to the confidence value when both views see the pixel (case 2).

B. Depth assignment

Given the trinocular matching cost function defined in Equation 4, the initial disparity of a given pixel p is determined by a typical local stereo matching method [41]. On the cost aggregation step, we adopt the edge-preserving guided filter [42] to filter computed cost maps with a support window of the size 9×9. The initial disparity d_p for a pixel p is determined by finding the disparity value with the minimal filtered cost, $d_p = \arg\min_d C_i^j(p,d)$. After disparity initialization, the method removes outlier disparity values in textureless and occluded regions with trinocular cross checking, and then uses another guided filter to propagate the disparity values of the reliable pixels to the unlabeled regions. Although the guided filter does not guarantee to construct the correct disparity values, its edge-preserving property fulfills the requirements for depth discontinuity alignment and depth smoothness mentioned at the beginning of this section. This way, we obtain the disparity map $\mathbf{D}_i^j$ for the fisheye image $\tilde{\mathbf{I}}_i^j$. Figure 7 demonstrates example results. Although the disparity maps are not particularly accurate, they provide sufficiently good depth samples for the view synthesis in our experiments. Finally, we convert the disparity map $\mathbf{D}_i^j$ into the depth map $\tilde{\mathbf{D}}_i^j$ by triangulation using Equation 2.

V. Stereoscopic Panorama Synthesis

As described in Section I, limited angular samples on the viewing circle introduces visible seams for synthesizing omnistereo panoramas. Although flow-based stitching methods can reduce the visible seams, they require dense views to work properly. In addition, they often generate ghosting with sparse views even if cross-view optical flows are estimated correctly. Our goal is to synthesize omnistereo panoramas without visible seams and ghosting. With the depth reconstruction method described in Section IV, we have eight rectified images $\tilde{\mathbf{I}}_i^j$ and their corresponding depth maps $\tilde{\mathbf{D}}_i^j$. Together with the camera parameters recovered from calibration (Section III-A), they define a 3D color point cloud depicting the scene captured by the proposed polycamera. Although it is possible to use traditional 3D warping methods [43] for panorama synthesis, they are very inefficient for the complicated omnistereo projection model. The paper proposes a dedicated 3D warping methods for efficient view synthesis. In Section V-A we review the omnistereo projection model. Section V-B introduces efficient projection formulations for forward and inverse mapping. Section V-C presents a rasterization-based backward ray tracing method which utilizes the efficient mappings.

A. The omnistereo projection model

We first describe the omnistereo model [10] for projecting a 3D point onto the two cylinders corresponding to the image planes respectively for the left and right eyes. We set the center of the polycamera as the origin of the world coordinate system for panorama synthesis. The center of the polycamera is defined as the center of mass of the four fisheye cameras. The z axis is defined along the viewing direction of the first fisheye camera $\mathbf{V}_0$ while the x–y plane is parallel to the image plane of $\mathbf{V}_0$.

The omnistereo model uses the circular projection in which the left-eye and right-eye images share the same cylindrical image plane, called *image cylinder*. The left and right eyes are located on an inner circle, called *viewing circle*. Without loss of generality, we assume the viewing circle locates on the *viewing plane*, $y = 0$. The viewing direction is along a tangent line to the viewing circle. Figure 9 depicts the configuration for the circular projection. The radius of the viewing circle r_v is given by the user, defining the separation between two eyes. The radius of the image cylinder r_c can be set by the user and its default value is the sum of the focal length f of the polycamera and the camera separation r, where f is the average focal length for the four fisheye cameras and r is the average distance of four fisheye cameras to the polycamera's center.

Given a 3D point X, its projections (x_L, x_R) on the left-eye and right-eye images are the intersections of the image cylinder and the 3D lines (l_L, l_R) which originate from their viewpoints (v_L, v_R) on the viewing circle and pass through X. The viewpoints can be found by taking the tangent lines

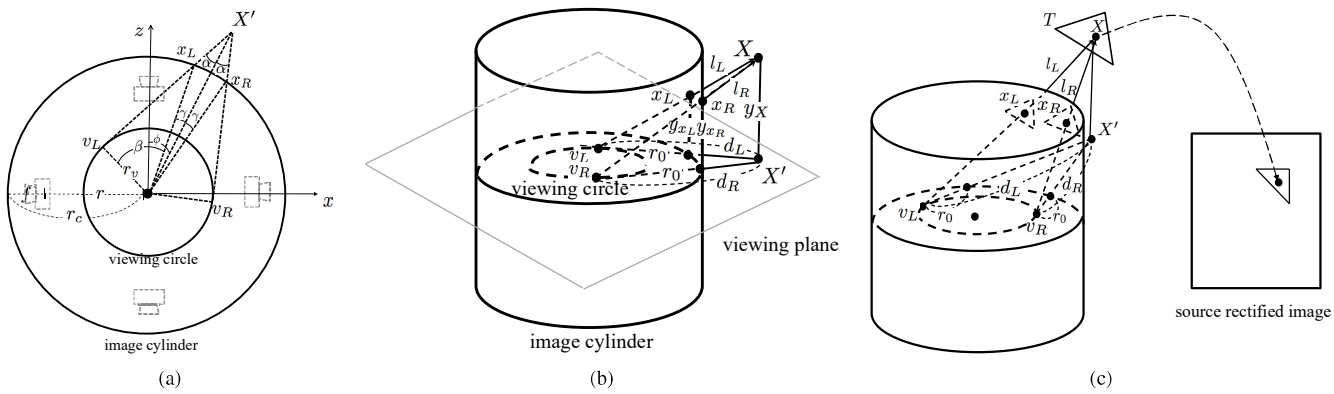


Fig. 9. The omnistereo projection model. (a) The configuration of the circular projection from the top view. This figure illustrates the angular relationship between the 3D point X, its projections (x_L, x_R), and the viewpoints (v_L, v_R). (b) A 3D view for the circular projection. It illustrates the y-scaling operation according to the ratio of the distances on the viewing plane. (c) The process of rasterization-based backward ray tracing.

of the viewing circle passing through X', the projected point of X on the viewing plane. We denote the circular projections for the left and right images as $x_L = \pi_L(X)$ and $x_R = \pi_R(X)$ respectively.

B. Forward and inverse mapping

Although it is possible to obtain the projections of a 3D point X by following the above procedure, it is computationally expensive for computing the viewpoints (v_L, v_R) and the 3D lines (l_L, l_R). This section presents formulations for forward and inverse mappings for more efficient computation.

Forward mapping. Given a 3D point X, by inspecting the circularly symmetric projections from the image cylinder to the viewing circle, it is possible to obtain its projections (x_L, x_R) without computing (v_L, v_R) and (l_L, l_R). First, a 3D point is represented with a cylindrical coordinate system, (r, ϕ, y) where (r, ϕ) is the polar coordinate of its projection on the viewing plane and y is its height above the viewing plane. Let $X = (r_X, \phi_X, y_X)$ where $r_X = |X'|$ and ϕ_X is its longitude (i.e., the angle between X' and the z-axis). Its projections $x_L = (r_c, \phi_{x_L}, y_{x_L})$ and $x_R = (r_c, \phi_{x_R}, y_{x_R})$ can be obtained directly by the following steps:

1) *Longitude shifting*. $\phi_{x_L} = (\phi_X - \gamma)$ and $\phi_{x_R} = (\phi_X + \gamma)$, where $\gamma = \pi/2 - \alpha - \beta$ is the longitude shift on the image cylinder, $\alpha = sin^{-1}(r_v/r_X)$ and $\beta = cos^{-1}(r_v/r_c)$. Figure 9(a) illustrates the longitude shifting.
2) *Y-scaling*. $y_{x_L} = y_{x_R} = s y_X$, where $s = r_0/\sqrt{r_X^2 - {r_v}^2}$, is the scaling factor and $r_0 = \sqrt{{r_c}^2 - {r_v}^2}$ is the fixed distance between all points on the image cylinder and their viewpoints on the viewing circle. Figure 9(b) illustrates the y-scaling step.

In the above steps, only α and s depend on X and need to be re-computed for each 3D point in forward mapping. Thus, the procedure can be very efficient.

Inverse mapping. As most image processing tasks, inverse mapping is used more frequently in practice. For our case, given a point x_L on the left panorama (or x_R on the right one), the inverse mapping attempts to find the 3D point X. However, different from the forward mapping, determining X requires to know the distance d_L (d_R) from the viewpoint v_L (v_R) to X. If the distance is unknown, the inverse mapping can only determine the viewpoint v_L and the viewing direction l_L for the given x_L (or v_R and l_R for x_R). We first determine v_L and l_L:

1) *Viewpoint computation.* From Figure 9(a), the longitude deviation from the projected point to the view point is β which was introduced in the first step of the forward mapping. Thus, we have

$$v_L = (r_v sin(\phi_{x_L} - \beta), 0, r_v cos(\phi_{x_L} - \beta)) \quad (8)$$
$$v_R = (r_v sin(\phi_{x_R} + \beta), 0, r_v cos(\phi_{x_R} + \beta)). \quad (9)$$

2) *Viewing direction computation.* Let $(l_x, 0, l_z)$ represent the unit vector of projected viewing line on the viewing plane. This vector is orthogonal to the viewpoint vector. Thus, for v_L, it can be expressed as follows:

$$(l_x, l_z) = (sin(\phi_{v_L} + \pi/2), cos(\phi_{v_L} + \pi/2)) \quad (10)$$
$$= (cos(\phi_{x_L} - \beta), -sin(\phi_{x_L} - \beta)). \quad (11)$$

Similarly, $(l_x, l_z) = (-cos(\phi_{x_R} + \beta), sin(\phi_{v_R} + \beta))$ for v_R.

3) *Determining X'*. If the distance d_L to X is known, the projection X' on the viewing plane can be determined as $X' = (v_x + d_L l_x, 0, v_z + d_L l_z)$, where (v_x, v_z) denotes the viewpoint v_L. X' can be determined similarly for x_R.
4) *Y-scaling*. For v_L, the height y_X of X can be determined by $y_X = s' y_{x_L}$ where $s' = d_L/r_0$. For v_R, y_X can be determined similarly. Together, X' and y_X give us the 3D point X.

C. Rasterization-based ray tracing

Given a 3D scene point cloud $\mathbf{X}$, with the circular projections π_L and π_R, a naive solution for stereoscopic panorama synthesis would be the forward mapping. The approach would however suffer from the problems that forward mapping often encounters, such as holes. Although the problem can be alleviated by splatting, it is usually difficult to set a proper kernel. Either holes or blur artifacts will appear in the result. A common remedy is to use the inverse mapping. We propose

a rasterization-based backward ray tracing method to compute the inverse mapping. In the following we describe how to synthesize the left panorama and the right one is obtained similarly.

First, we convert the 3D point cloud $\mathbf{X}$ into 3D meshes $\mathbf{M}$ by triangulation along with the image grid of the rectified images. For each triangle T in $\mathbf{M}$, we project its three vertices onto the image cylinder using the forward mapping of π_L, and then obtain T's projection T' on the image cylinder. We then rasterized the triangle T' on the image space of the image cylinder. For each rasterized pixel x_L we compute its ray intersection of its projection line l_L to the 3D triangle T. Finally, the color of x_L is assigned with the corresponding source pixels which are found by projecting the intersected 3D point back to the source rectified images. Figure 9(c) illustrates the process. Since the rasterization process is time consuming, we combine the inverse mapping method for more effective ray intersection computation for each rasterized pixel to reduce the overall computing cost.

Assume that the plane equation of the triangle T is $n_x x + n_y y + n_z z = d_0$. Given a rasterized pixel x_L on the left panorama, we first apply the first two steps of the inverse mapping described in Section V-B to obtain its viewpoint $(v_x, 0, v_z)$ and viewing direction $(l_x, 0, l_z)$. By combining them with the plane equation, we have $n_x(v_x + l_x d_L) + n_y(s' y_{x_L}) + n_z(v_z + l_z d_L) = d_0$. Therefore, the distance d_L can be determined as

$$d_L = \frac{d_0 - (n_x v_x + n_z v_z)}{(n_y/r_0) y_{x_L} + (n_x l_x + n_z l_z)}. \tag{12}$$

Once d_L is determined, we can apply the last two steps of the inverse mapping to find the corresponding 3D point X. The procedure is similar for x_R.

Collision Handling. It is possible that a pixel on the image cylinder is covered by more than one candidate if its corresponding point is visible in multiple cameras. One way to resolve it is to compare the candidates' depth values and take the color of the closest depth. However, we found that the depth value could be unreliable if the candidate is on the silhouette of some object. Because of the depth discrepancy between the object and its background, the corresponding triangle could be slanted and the depth value varies quickly within the triangle. We check the reliability by depth consistency. If the candidate's depth (obtained by ray tracing described above) is too different from the depth value calculated by bilinear interpolation on the source pixels of the depth map, corresponding to the three vertices of the triangle T, it is without depth consistency and discarded.

Color Blending. Even after removing candidates without depth consistency, it is possible that a pixel on the image cylinder is still covered by more than one plausible candidate. All these plausible candidates are blended by weights. The weight of the candidate is defined similarly to the pixel confidence defined in Section IV-A as $e^{-\alpha_p}$ where α_p is the incident angle between the candidate and the optical axis. This way, the candidates locating closer to the image center have higher weights than the ones closer to the image boundary.

VI. Experimental Results

This section evaluates the proposed method, compares it with alternative methods and presents several stereoscopic 360° images captured with the proposed polycamera and synthesized with the proposed method. Using the prototype polycamera described in Section I, each captured ployview image contains four fisheye images of resolution 4000×3000. In the following, we provide evaluation for the three main components of the proposed system on calibration, depth estimation and view synthesis.

Calibration and rectification. We report the reprojection error of the chessboard corners to evaluate the proposed calibration method. The root mean square and the maximum of the corner reprojection errors are 1.028 and 4.82 (pixels), respectively. For quantitative evaluation for rectification, we use horizontal alignment error of chessboard corners to measure the performance of rectification. The root mean square and the maximum of the alignment errors are 0.47 and 2.68 (pixels), respectively. We also present the average FOV of fisheye lenses and the extrinsic manufacture error since the parameters influence the overall stereo viewing coverage of the proposed camera, as discussed in Section III-B. The average FOV of fisheye lenses is up to $200° \times 200°$, larger than that of the lens specification ($190° \times 190°$). Let $\mathbf{R}'_{i,i+1}$ and $\mathbf{T}'_{i,i+1}$ represent the ideal value of $\mathbf{R}_{i,i+1}$ and $\mathbf{T}_{i,i+1}$, respectively. The translation error is computed by $|\mathbf{T}_{i,i+1} - \mathbf{T}'_{i,i+1}|/|\mathbf{T}'_{i,i+1}|$. The rotation error is computed by $|Rod(\mathbf{R}^T_{i,i+1}\mathbf{R}'_{i,i+1})|/|Rod(\mathbf{R}'_{i,i+1})|$, where $Rod(.)$ is a Rodrigues vector representation[3]. The average rotation error and translation error are 1.74% and 5.43%, respectively.

Panoramic depth estimation. We compare different methods for matching cost computation, including binocular matching cost (BM), trinocular matching cost (TM) and trinocular matching cost with adaptive weights (TMAW). The BM cost only uses two views and includes the term $\rho_{i,j}(p, d)$ defined in Equation 4. The TM cost sets both $w(p, d) = 1$ and $w(p^c, d^c) = 1$ in Equation 4. The TMAW cost is defined as Equation 4. For quantitative analysis, 15 polyview images were processed and their average consistent matching rates are reported. The consistent matching rates for BM, TM, and TMAW are 58%, 67%, and 70%, respectively. TM increases nearly 10% consistent matching rate, and TMAW provides even more improvement. Figure 10 compares the estimated disparity maps using two examples. Qualitatively, the methods with the higher consistent matching rates tend to achieve better depth smoothness and more reliable depth estimation for the areas close to the borders and the corners of the image. Figure 11 shows the impact of cross-view depth consistency on the quality of view synthesis. The panoramic depth maps were synthesized with the same way as that for panorama synthesis. It is clear that TMAW better preserves structures and details.

We experimented with different fisheye FOV specifications and observed the impact on depth estimation. We generated three sets of rectified images with 180°, 190°, and 200° fisheye FOVs. The average consistent matching rates for trinocularly

[3]A vector representation for a rotation matrix, where the normalized vector represents the rotation axis and the magnitude represents the rotation angle.

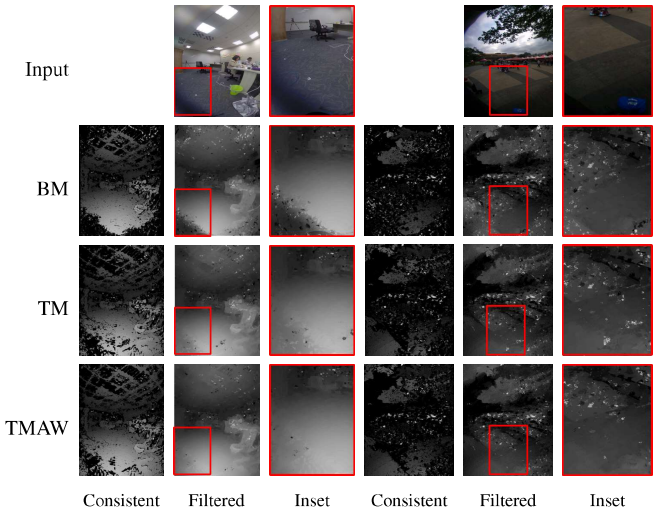


Fig. 10. Comparisons of different matching costs, BM (binocular matching cost), TM (trinocular matching cost), and TMAW (trinocular matching cost with adaptive weights). Two examples are shown here. The left example is an indoor scene at an office and the right one is an outdoor scene. For each example, we show the input and the results of BM, TM and TMAW. For each result, we show the disparity map after consistent matching, the filtered disparity map and an inset to highlight the problematic area in that order. For the indoor example, the consistent matching rates are BM (54%), TM (57%) and TMAW (62%). For the outdoor example, they are BM (39%), TM (49%) and TMAW (53%). This example is particularly challenging for BM since the pixels close to the image boundary are very dark due to vignetting.

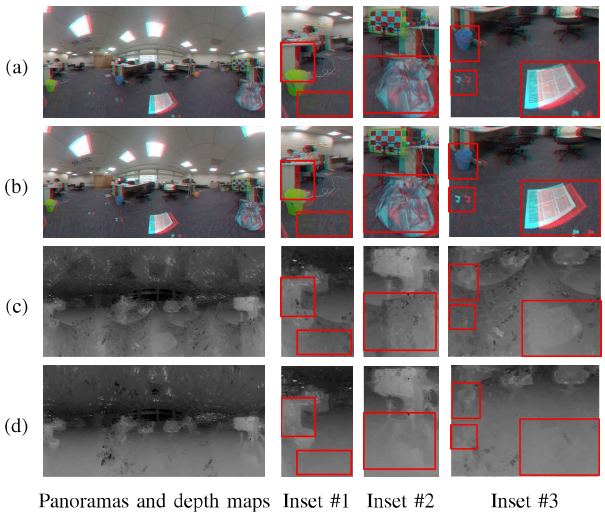


Fig. 11. Comparisons of BM and TMAW on synthesized stereoscopic panoramas and depth maps. (a)(c) The stereoscopic panorama and the panoramic depth map synthesized with binocular matching. (b)(d) The stereoscopic panorama and the panoramic depth map synthesized with the proposed trinocular matching with adaptive weighting. The depth map of TMAW shows much better depth consistency cross input views. The stereoscopic panorama with TMAW also preserves structures and textures better as outlined with red boxes in the insets. Results of BM exhibit shape distortion and blurriness.

visible regions are 60.7%, 63.86%, and 64.19%, respectively. A larger fisheye FOV improves the consistent matching rate, but the gain from 190° to 200° is not significant. Since larger FOVs incur more computation, we settled at 190° for a good compromise.

Stereoscopic panorama synthesis. We compare the proposed panorama synthesis method with a few alternatives. We implemenetd the flow-based interpolation method [14] in a forward setting. The rectified images are dewarped to cylinder images and flows between cylinder images are computed from the estimated disparity maps. A small radius (1.4mm) is set for the viewing circle to reduce holes and widths of the holes are at most 15 pixels for this setting. The Navier-Stokes-based inpainting method [44] is used for filling holes for both the flow-based method and our 3D forward warping. Figure 12 compares the flow-based interpolation method and the proposed 3D forward warping method. From the inset (Figure 12(c)), it is clear that the flow-based method exposes obvious ghosting on close-by objects with large flows because the interpolation does not account for depth variations. Our 3D forward warping does not suffer from ghosting here.

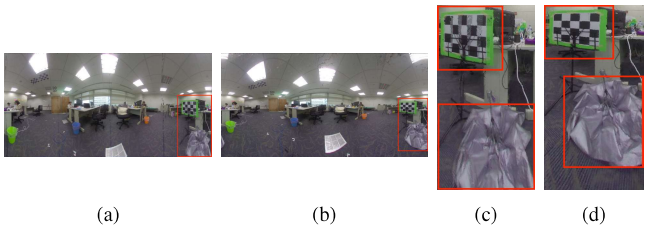


Fig. 12. Comparisons of flow-based interpolation and 3D forward warping on the example *Office*. (a)(c) The result of flow-based interpolation and an inset. (b) The result of 3D forward warping and an inset. The result of flow-based stitching exposes ghosting on close-by objects with large flows, as shown in the inset.

Figure 13 compares the proposed 3D forward warping approach and the proposed 3D backward warping method. As discussed in Section V-B, the forward warping method suffers from the problem with holes. Although the problem can be alleviated by inpainting, it is clear from Figure 13 that the results of forward warping still exhibit artifacts of irregular noises and distorted structures even with inpainting.

Next, we compare the strategy of taking the candidate closest to the camera and the proposed strategy by checking with depth consistency as discussed in Section V-C. Figure 14 shows an example for the comparison. By considering depth consistency, unreliable candidates are removed. It is especially effective on the cross-view regions, where source pixels are around boundaries of the rectified images.

Finally, we compare the color blending with equal weights and weighting by the incident angles. From Figure 15, it is clear that the color transition is much smoother by using the proposed weighting scheme.

We have taken the prototype polycamera to capture several scenes. To test the versatility of the proposed camera and methods, we explored scenes with different characteristics including outdoor scenes with large depth ranges, indoor scenes with significant depth discontinuities and sequences with moving objects. Figure 16-Figure 18 show four examples of the captured fisheye images, the synthesized views and the stereoscopic 360° panoramas in the form of anaglyph 3D images. More examples can be found in the supplementary material which also includes images that are ready for viewing using VR displays. Figure 16 shows an outdoor example with

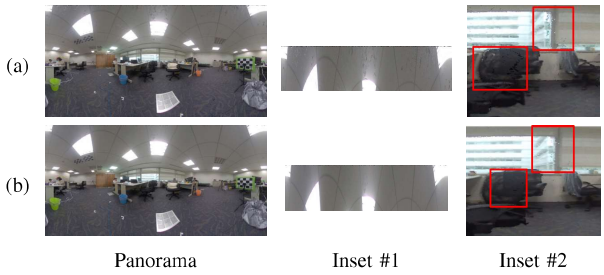


Fig. 13. Comparisons of forward warping and backward warping on the example *Office*. (a) The result of forward warping. (b) The result of backward warping. The result of forward warping usually have holes, present aliasing artifacts, produce irregular lines and contain noise particularly around object boundary as shown in inset #1 and outlined with red boxes in inset #2.

Fig. 14. Comparisons of resolving colliding candidates with the closest depth and with depth consistency on the example *Software Park*. (a)(c) The panorama of resolving with the closest depth and its inset. (b)(d) The result of resolving with depth consistency and its inset. The result with depth consistency maintain the structure better and have less artifacts as outlined with red boxes in the insets.

complicated depth structures. There is a big tree with complicate silhouettes and depth variations. The complicated shapes of the branches and leaves often present great challenges to depth estimation methods. Figure 17 demonstrates another outdoor example where there is significant depth variation. There is a building far away and there are several bushes closer to the camera. Our method handles both outdoor scenes quite effectively. Figure 18 shows a scene inside the hall of Grand hotel. There are occasionally visual artifacts due to inaccurate depth estimates, but in general when viewing with head-mounted displays, viewers can enjoy exploring the captured scenes freely without noticing them and the stereoscopy offers more vivid viewing experiences by enhancing the depth perception.

User study. We conducted a user study for evaluating the flow-based interpolation method, our forward warping method, and our backward warping method to determine which method delivers the best user experience when viewing with VR displays.

Fig. 15. Comparisons of color blending with equal weights and adaptive weights by the incident angles for view synthesis. This figure shows two examples. (a) The panorama of blending with equal weights and its inset. (b) The panorama with adaptive weights and its inset. It is clear that the color transition is much smoother when using adaptive weights.

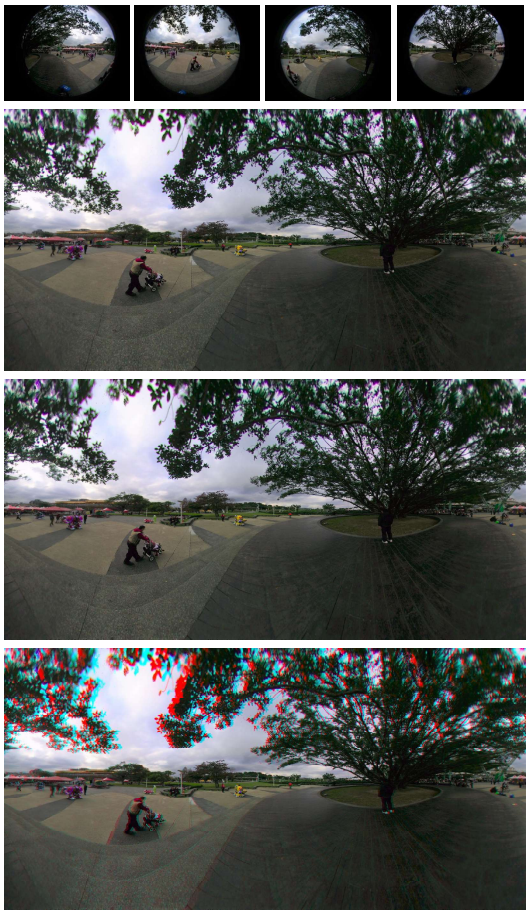


Fig. 16. The example *Flower Museum*. An outdoor scene with a big tree at the center. The branches and leaves present challenges for depth estimation. From the top to the bottom, they are the input fisheye images, the synthesized panoramas for the left and right eyes, and the stereoscopic 360° panorama in the anaglyph form.

18 adults participated the user study, with ages from 25 to 55 years old. Six scenes were used in the study, including *Office* (Figure 12), *Grand Hotel Inside* (Figure 18), *Grand Hotel Outside* (Figure 17), *Flower Museum* (Figure 16), *Software Park* (Figure 14), and *Library* (Figure 1). The participants viewed results of the three methods using VR displays in a random order. For each result, a participant had three minutes to explore the scene. After seeing all three results for a scene, two questions were asked:

1) Which one delivers the best image quality?
2) Which one delivers the best stereoscopic perception?

They could also comment freely on the results.

Figure 20(a) shows the results. In terms of image quality, 20%, 11%, and 69% of participants favored the results of flow-based interpolation, 3D forward warping, and 3D backward warping, respectively. The 3D backward warping method was the favorite for each of the six scenes. It is worth noting that flow-based interpolation is more favored than 3D forward warping. From participants' comments, this is because the irregular noises of the 3D forward warping method at high-latitude regions are very disturbing. In addition, the flow-based

Fig. 17. The example *Grand Hotel Outside*. An outdoor scene with a building at a distance and several brushes near the camera. From the top to the bottom, they are the input fisheye images, the synthesized panoramas for the left and right eyes, and the stereoscopic 360° panorama in the anaglyph form.

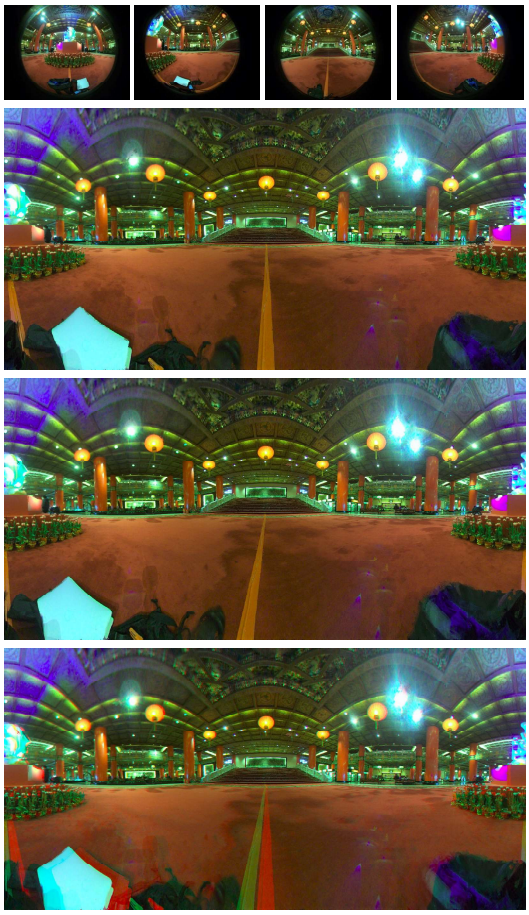


Fig. 18. The example *Grand Hotel Inside*. The hall of Grand Hotel. From the top to the bottom, they are the input fisheye images, the synthesized panoramas for the left and right eyes, and the stereoscopic 360° panorama in the anaglyph form.

interpolation method performs poorly for scenes with many close-by objects and occlusions such as *Office*.

As for stereoscopic perception, 22%, 36%, and 42% of participants favored the results of flow-based interpolation, 3D forward warping, and 3D backward warping, respectively. Although 3D backward warping is the favorite, participants reflected that the stereoscopic perception of the results of forward warping is very close to that of the results of backward warping. We note that flow-based interpolation had certain votes (22%) despite of its small disparity values. We suspect that it is because of vertical scaling [14] on the scene. Thus, the size of the main region in its result is larger than those in 3D warping methods.

To verify this assumption, we generate another set of results by decreasing the vertical FOVs of omnistereo panoramas of 3D warping methods while maintaining the same image size. The vertical scale of the main region becomes roughly the same as that in flow-based interpolation, as shown in Figure 19. Another user study was conducted with vertical scaling and the results are shown in Figure 20(b). This time, flow-based interpolation cannot take advantage of the scaling of the main region and had much lower votes. The votes divide as: flow-based interpolation (13%), 3D forward warping (30%) and 3D backward warping (57%). At the same time, the votes of image quality also change: flow-based interpolation (14%), 3D forward warping (8%) and 3D backward warping (78%). The user study shows that users pay more attention to the main region.

From the user study, we conclude that 3D backward warping generates the best results. In addition, most participants were not aware of boundaries between input views in the results of 3D warping. Only one noticed a little depth discontinuity. Additionally, none reported perceiving depth differences within textureless regions. It shows that the proposed trinocular matching method achieves good cross-view depth consistency and sufficient depth smoothness.

Runtime performance. The current system was implemented using C++. The proposed backward warping method for panorama synthesis was implemented with OpenGL shading language (GLSL) to take advantage of GPUs. All experiments were performed on a laptop with an Intel core i7 2.40Hz processor and an Intel HD Graphics 5500 GPU. Table II shows the runtime of camera calibration, image rectification, depth reconstruction, and panorama synthesis. The nonlinear

(a) (b) (c)

Fig. 19. Vertical scaling. (a) The result of flow-based interpolation with vertical FOV 120°. (b) The result of 3D backward warping with vertical FOV 120°. (c) The vertically scaled result of (b). The vertical FOV is reduced to 105°. The vertical size of the main region (i.e., the region between the yellow lines) is now roughly the same as (a).

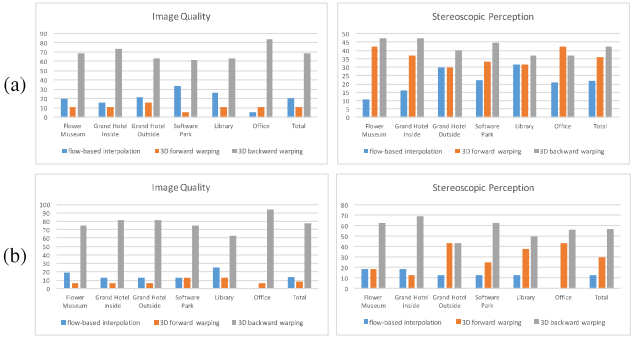


Fig. 20. The results of the user study on image quality (the left column) and stereoscopic perception (the right column) without vertical scaling (a) and with vertical scaling (b).

Preprocessing	
Operation	Runtime (seconds)
Camera Calibration	16
Image Rectification	431
Processing per VR scene	
Operation	Runtime (seconds)
Depth Reconstruction	320
Panorama Synthesis (with GPU)	65

TABLE II
PROCESSING TIME OF THE PROPOSED METHODS.

(a) (b) (c) (d)

Fig. 21. Limitations. (a) Aliasing around object boundaries. (b) The protruding background. (c) The line distortions. (d) The ghosting artifacts around the corners.

optimization for camera calibration converged within few iterations and took only 16 seconds. Image rectification includes view overlap computation, spherical rectification, and image remapping. It totally spent 431 seconds. Note that the image rectification is scene independent and therefore can be pre-computed. The depth reconstruction took about 320 seconds for eight rectified images. Finally, stereoscopic panorama synthesis took 64 seconds for generating two panoramas of the resolution 5440×2720. As a reference, the CPU version would take 1,578 seconds for the same resolution.

Limitations. The user study reveals a few limitations of the proposed method. Some participants noticed aliasing around the boundaries of close-by objects. Although the depth estimation uses edge-aware guided filters, some occlusion boundaries still cannot be accurately extracted. Also, if a background is textureless and surrounded by the foreground, participants could feel that the background regions is merged into the foreground and become protruding. Although not reported by participants, there are some other limitations. For thin lines close to the camera, the depth estimation error could result in visible line distortion. In addition, for objects very close to the camera, the proposed method could fail to recover the large occluded region and there could be ghosting around the object boundary, particularly when the objects are close to the top or the bottom of the viewing sphere. Figure 21 shows examples illustrating these limitations.

VII. Conclusion and Future Work

This paper proposes a camera system and a set of methods for synthesizing stereoscopic 360° panoramas. The camera consists of only four cameras with fisheye lens, making the camera portable and inexpensive. The reduction on the number of views brings up challenges for panorama synthesis. The paper addresses these challenges with methods for polyview rectification, panoramic depth estimation and view synthesis. The synthesized stereoscopic 360° panoramas allow viewers to explore the captured scenes freely using VR displays.

In the future, we would like to explore more design options. For example, it is interesting to know whether it is possible to further reduce the number of cameras by using the ones with even wider fields of view. Another interesting question would be how to increase the number of cameras to have more view overlaps but still maintain good portability. On the algorithm side, the depth estimation could benefit from trilateral filtering [45] for better preserving depth discontinuities while maintaining computation efficiency. Deformable spheres [46] guided by spare 3D points could be an effective alternative to dense depth reconstruction for omnistereo panorama synthesis, and we have obtained preliminary but promising results [47].

References

[1] "Google Cardboard," https://vr.google.com/cardboard/.
[2] "HTC vive," https://www.vive.com/tw/.
[3] "Oculus," https://www.oculus.com/.
[4] S. B. Kang, M. Wu, Y. Li, and H. Shum, "Large environment rendering using plenoptic primitives," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1064–1073, 2003.
[5] "Ptgui," https://www.ptgui.com/.
[6] "Hugin: Panorama photo stitcher," http://hugin.sourceforge.net/.
[7] Y. Xu, Q. Zhou, L. Gong, M. Zhu, X. Ding, and R. K. F. Teng, "High-speed simultaneous image distortion correction transformations for a multicamera cylindrical panorama real-time video system using FPGA," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 6, pp. 1061–1069, 2014.
[8] "360 camera online," http://360cameraonline.com/known-360-cameras/.
[9] H.-C. Huang and Y.-P. Hung, "Panoramic stereo imaging system with automatic disparity warping and seaming," *Graphical Models and Image Processing*, vol. 60, no. 3, pp. 196–208, 1998.

[10] S. Peleg, M. Ben-Ezra, and Y. Pritch, "Omnistereo: Panoramic stereo imaging," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 279–290, 2001.

[11] Y. Pritch, M. Ben-Ezra, and S. Roy, "Optics for omnistereo imaging," in *In Foundations of Image Understanding (IFIU 2001)*, 2001, pp. 447–467.

[12] K. Tanaka and S. Tachi, "TORNADO: Omnistereo video imaging with rotating optics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 6, pp. 614–625, 2005.

[13] A. S. Amini, M. Varshosaz, and M. Saadatseresht, "Evaluating a new stereo panorama system based on stereo cameras," *International Journal of Scientific Research in Inventions and New Ideas*, 2014.

[14] R. Anderson, D. Gallup, J. T. Barron, J. Kontkanen, N. Snavely, C. Hernández, S. Agarwal, and S. M. Seitz, "Jump: Virtual reality video," in *Proceedings of ACM SIGGRAPH Asia 2016*, 2016, pp. 198:1 – 198:13.

[15] "Google Jump," https://vr.google.com/jump/.

[16] "Facebook Surround 360," https://facebook360.fb.com/facebook-surround-360/.

[17] R. Swaminathan and S. Nayar, "Non-Metric Calibration of Wide-Angle Lenses and Polycameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1172 – 1178, 2000.

[18] S. Tzavidas and A. K. Katsaggelos, "A multicamera setup for generating stereo panoramic video," *IEEE Transaction on Multimedia*, vol. 7, no. 5, pp. 880 – 890, 2005.

[19] C. Richardt, Y. Pritch, H. Zimmer, and A. Sorkine-Hornung, "Megastereo: Constructing high-resolution stereo panoramas," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR 2013)*, 2013, pp. 1256–1263.

[20] R. Aggarwal, A. Vohra, and A. M. Namboodiri, "Panoramic stereo video with a single camera," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, 2016.

[21] "Fraunhofer hhi omnicam 360," https://www.hhi.fraunhofer.de/en/departments/vit/technologies-and-solutions/capture/panoramic-uhd-video/omnicam-360.html.

[22] V. C. Couture and S. Roy, "The omnipolar camera: A new approach to stereo immersive capture," in *Proceedings of IEEE International Conference on Computational Photography (ICCP 2013)*, 2013, pp. 1–9.

[23] K. Matzen, M. F. Cohen, B. Evans, J. Kopf, and R. Szeliski, "Low-cost 360 stereo photography and video capture," *ACM transactions on Graphics*, vol. 36, no. 4, pp. 148:1–148:12, 2017.

[24] R. Konrad, D. G. Dansereau, A. Masood, and G. Wetzstein, "Spinvr: Towards live-streaming 3d virtual reality video," *ACM transactions on Graphics*, vol. 36, no. 6, pp. 209:1–209:12, 2017.

[25] D. Schneider and H.-G. Maas, "Geometric modelling and calibration of a high resolution panoramic camera," *Optical 3D Measurement Techniques*, vol. II, pp. 122–129, 2003.

[26] J. A. Parian and A. Gruen, "Panoramic camera calibration using 3D straight lines," *Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XXXVI, 2005.

[27] X. Y. Guan, L.-K. Shark, G. Hall, and W. Deng, "Calibration of rotating line spherical camera based on checkerboard pattern on multiple planes and its accuracy assessment," in *Proceedings of the UK postgraduate workshop of British Machine Vision Conference (BMVC 2010/)*, 2010, pp. 8.1–8.10, 8.

[28] "Spheron," https://www.spheron.com/home.html.

[29] S. K. Nayar, "Sphereo: Recovering depth using a single camera and two specular spheres," in *Proceedings of SPIE Conference on Optics, Illumination, and Image Sensing for Machine Vision (SPIE 1988)*, 1988, pp. 245–254.

[30] D. Southwell, A. Basu, M. Fiala, and J. Reyda, "Panoramic stereo," in *Proceedings of the International Conference on Pattern Recognition (ICPR 1996)*, 1996.

[31] S. Nene, "Stereo with mirrors," in *Proceedings of IEEE International Conference on Computer Vision (ICCV 1998)*, 1998, pp. 1087–1094.

[32] J. Cluckman, S. Nayar, and K. Thorek, "Real-time omnidirectional and panoramic stereo," in *Proceedings of Defense Advanced Research Projects Agency, Image Understanding Workshop (DARPA 1998)*, 1998, pp. 299–303.

[33] Z. Arican and P. Frossard, "Dense disparity estimation from omnidirectional images," in *Proceedings of the IEEE International Conference on Advanced Video and Signal based Surveillance (ICAVS 2007)*, 2007.

[34] Z. Lee and T. Q. Nguyen, "Multi-resolution disparity processing and fusion for large high-resolution stereo image," *IEEE Transactions on Multimedia*, vol. 17, no. 6, pp. 792–803, 2015.

[35] M. Schönbein and A. Geiger, "Omnidirectional 3d reconstruction in augmented manhattan worlds," in *Proceedings of International Conference on Intelligent Robots and Systems (IROS 2014)*, 2014.

[36] "Nokia ozo," https://ozo.nokia.com.

[37] "Samsung gear 360," http://www.samsung.com/global/galaxy/gear-360/.

[38] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1335–1340, 2006.

[39] J. Fujiki, A. Torii, and S. Akaho, "Epipolar geometry via rectification of spherical images," in *Proceedings of International conference on Computer vision/computer graphics collaboration techniques*, 2007, pp. 461–471.

[40] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proceedings of the Third European Conference on Computer Vision (ECCV 1994)*, 1994, pp. 151 – 158.

[41] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 7 – 42, 2002.

[42] K. He, J. Sun, and X. Tang, "Guided image filtering," in *Proceedings of the European Conference on Computer Vision and Pattern Recognition (ECCV 2010)*, 2010, pp. 1–14.

[43] L. McMillan, "An image-based approach to three-dimensional computer graphics," University of North Carolina at Chapel Hill, Tech. Rep., 1997.

[44] M. Bertalmo, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting." in *Proceedings of the International Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, 2001, pp. 355–362.

[45] D. Chen, M. Ardabilian, and L. Chen, "A Fast Trilateral Filter based Adaptive Support Weight Method for Stereo Matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. PP, no. 99, pp. 1–14, 2014.

[46] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh, "Rich360: Optimized spherical representation from structured panoramic camera arrays," *ACM Transaction on Graphics*, vol. 35, no. 4, pp. 63:1–63:11, 2016.

[47] S.-K. Huang, H.-S. Lin, and M. Ouhyoung, "Effective omnistereo panorama video generation by deformable spheres," in *ACM SIGGRAPH 2017 Posters*, 2017, pp. 22:1–22:2.