Low-Cost 360 Stereo Photography and Video Capture

KEVIN MATZEN, MICHAEL F. COHEN, BRYCE EVANS, JOHANNES KOPF, and RICHARD SZELISKI,
Facebook Inc.

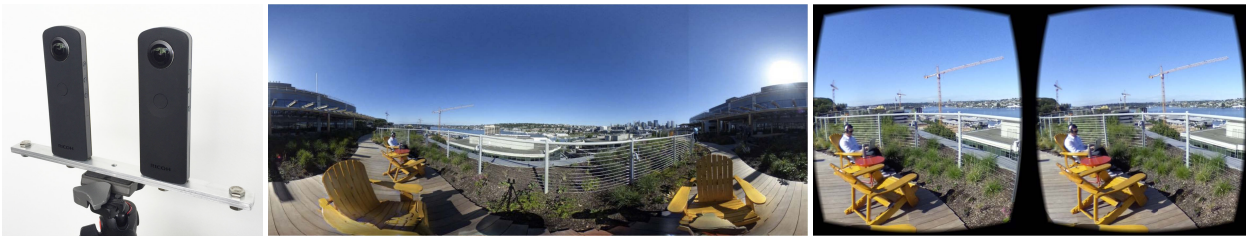

Fig. 1. (a) our stereo rig; (b) left eye equirectangular image; (c) cross-eyed stereo pair from balcony scene.

A number of consumer-grade spherical cameras have recently appeared, enabling affordable monoscopic VR content creation in the form of full 360° × 180° spherical panoramic photos and videos. While monoscopic content is certainly engaging, it fails to leverage a main aspect of VR HMDs, namely stereoscopic display. Recent stereoscopic capture rigs involve placing many cameras in a ring and synthesizing an omni-directional stereo panorama enabling a user to look around to explore the scene in stereo. In this work, we describe a method that takes images from two 360° spherical cameras and synthesizes an omni-directional stereo panorama with stereo in all directions. Our proposed method has a lower equipment cost than camera-ring alternatives, can be assembled with currently available off-the-shelf equipment, and is relatively small and light-weight compared to the alternatives. We validate our method by generating both stills and videos. We have conducted a user study to better understand what kinds of geometric processing are necessary for a pleasant viewing experience. We also discuss several algorithmic variations, each with their own time and quality trade-offs.

CCS Concepts: • **Computing methodologies** → **Computational photography**; **Image processing**; **Image-based rendering**; **Virtual reality**;

Additional Key Words and Phrases: image stitching, panoramas, virtual reality, stereo

ACM Reference format:
Kevin Matzen, Michael F. Cohen, Bryce Evans, Johannes Kopf, and Richard Szeliski. 2017. Low-Cost 360 Stereo Photography and Video Capture. *ACM Trans. Graph.* 36, 4, Article 148 (July 2017), 12 pages.
https://doi.org/10.1145/3072959.3073645

1 INTRODUCTION

Over the last couple of years, the general public has gained access to low-cost virtual reality head mounted displays capable of delivering immersive experiences. These devices range in price from hundreds of dollars for dedicated PC headsets down to just a few dollars for smartphone-based viewers. At the same time, fully-spherical cameras capable of capturing 360° × 180° content using fisheye lenses have found an audience among photography enthusiasts as well as regular consumers. Devices such as the Ricoh Theta and the Samsung Gear 360 have enabled lightweight capture, sharing, and interactive viewing of monoscopic 360° panoramas on sites such as Facebook and YouTube. However, monoscopic panoramas fail to leverage one of the primary advantages of VR HMDs, namely their stereoscopic display.

Professional content producers can deliver high-quality, stereo panoramic stills and videos using complicated and expensive hardware as well as costly processing. Some notable examples include the Google Jump and Facebook Surround 360 cameras. By combining multiple video cameras into a ring-shaped configuration, an omni-directional stereo panorama can be stitched by interpolating rays between cameras. These devices are priced such that they are accessible to professionals, but not to regular consumers.

A lower-cost approach is to have a user sweep an arc out with their smartphone, thus simulating a camera ring, which is the core principle behind the Google Cardboard Camera. However, this process can be tedious, video cannot be supported, and scenes with motion pose a challenge.

In this paper, we seek the simplest and cheapest solution to producing omni-directional stereo stills and videos with existing consumer devices. When it comes to conventional stereo photography viewed from the original viewpoint, the simplest such configuration uses two pinhole cameras displaced by a fixed baseline corresponding to a human inter-pupillary distance, with each image displayed to each eye. The question we explore is whether a two-camera configuration can be used to enable an immersive three degrees of freedom viewing experience.

A conventional camera does not satisfy the requirement that we be able to turn our heads 360°. Even an extremely wide angle lens will not allow the viewer to turn and look behind themselves. Fortunately, we can make use of two spherical cameras to satisfy this requirement. However, there are several technical challenges that prevent us from naively mapping one eye to each camera. Figure 2 illustrates several of these:

(1) There is no placement of the cameras to provide proper stereo in all directions. Any placement results in varying

0730-0301/2017/7-ART148 $15.00
https://doi.org/10.1145/3072959.3073645

apparent inter-occular distance depending on the viewing angle.
(2) Each camera is visible in the other view.
(3) Unless care is taken, the two views may not be oriented in the same direction.
(4) Each camera (or component of a single camera) may capture the scene with different exposure and/or gain.
(5) Spherical cameras may distort the imagery to compensate for the parallax caused by the two fisheye lenses not sharing a common focal point.
(6) Timing variations make absolute synchronization of frames difficult.
(7) It is impossible to simulate proper stereo in all directions with only two images, as is also the case with the omnidirectional stereo (ODS) projection.

While proper stereo cannot be simulated in all directions (particularly away from the horizon) with the omnidirectional stereo (ODS) two-panorama projection [Anderson et al. 2016], it is a common and widely supported format produced by devices such as the Google Jump and Facebook Surround 360 systems that can be viewed in most VR headsets as well as YouTube.

In this paper, we address the technical challenges in synthesizing an omni-directional stereo panorama from two fixed spherical cameras. Section 2 reviews prior work in the space of VR capture and stereo content display. Section 3 gives a brief overview of our hardware configuration and our algorithmic pipeline. Section 4 discusses the calibration necessary to robustly integrate measurements from consumer 360 devices that have geometric and photometric distortions. Section 5 describes how the rays captured by two spherical cameras can be warped to produce an omni-directional stereo panoramic projection. Section 6 describes how we estimate horizontal and vertical disparities (correspondences) between the two input images using either optical flow or rectified stereo matching. Section 7 presents our experiments, including the datasets we captured, our experiments comparing correspondence algorithms, and the results of our user study. We conclude in Section 8 with a discussion of our results and a list of potential future extensions.

2 RELATED WORK

Head-mounted displays have a long history dating back to the mid-twentieth century. Confined mostly to military applications, they entered more mainstream applications only recently. We now find that such devices are reaching the general public. At the same time, we are seeing new means of creating content emerge at a rapid pace. Here, we briefly touch on methods for photographic capture of 360° monoscopic and stereoscopic imagery. For a more detailed recent literature review, please see Anderson et al. [2016].

2.1 Panorama construction from moving cameras

There has been a great deal of work in constructing and viewing panoramas over the past few decades. Much of the early work relied on stitching together overlapping images taken from a moving camera [Szeliski and Shum 1997]. This technology was expanded to very large sets of images resulting in gigapixel panoramas [Kopf et al. 2007].

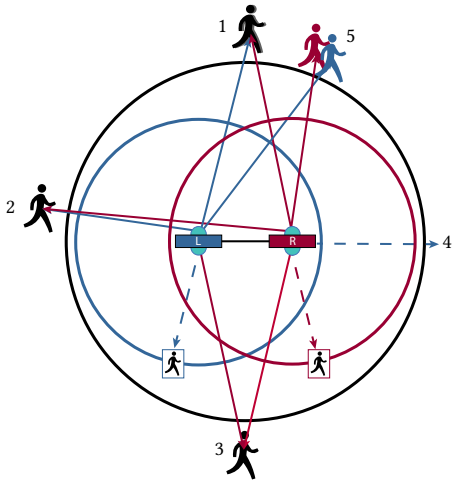

Fig. 2. Problems associated with using two side-by-side spherical cameras to create a full stereo experience. The diagram shows two cameras L (blue) and R (red) capturing two spherical images in corresponding colors. The problems: the apparent inter-occular distance changes from normal (1) to almost zero (2) to actually inverted since looking backwards reverses the eyes (3). The right camera obscures the left camera's view, and vice versa (4). The two cameras will likely be out of alignment (5), and the two cameras may have different exposures or have slight time offsets.

Capturing panoramas that include the ability to display stereo goes back to the early systems for capturing *omnidirectional stereo* (ODS) from different vertical slices taken from a rotating camera (or mirror) [Ishiguro et al. 1990; Peleg et al. 2001; Richardt et al. 2013; Tanaka and Tachi 2005]. Keeping a larger collection of images from rotating offset cameras [Debevec et al. 2015; Shum and He 1999] enables capturing a partial lightfield [Gortler et al. 1996; Levoy and Hanrahan 1996], which then allows the user to change their viewing position as well as orientation. Researchers have also described how to use a pair of spherical linescan cameras arranged vertically for 3D scene reconstruction [Kim and Hilton 2013], and how to stitch multiple overlapping stereo pairs into a stereo panorama [Zhang and Liu 2015].

2.2 Video camera arrays

Another line of work uses multiple video cameras stitched together to produce monoscopic [Lee et al. 2016; Nokia 2016; Perazzi et al. 2015] or stereoscopic [Anderson et al. 2016; Facebook 2016; Weissig et al. 2012] panoramic videos. Curved or folded mirrors can also be used to construct single-camera systems that can produce monoscopic [Nayar 1997] or stereoscopic [Aggarwal et al. 2016] panoramic videos, although the latter system cannot currently be manufactured at high enough tolerances to produce usable images.

The past two years have seen the introduction of a number of small form-factor consumer video capable panoramic cameras, such as the Ricoh Theta S and Samsung Gear 360 [Grayson 2016]. We use the Ricoh Theta S in our experiments. It retails for about $300 and has two fisheye lenses that each capture roughly a hemisphere to create full 360° × 180° spherical images.

The past year has also seen the introduction of small form-factor rigs using multiple stereo pairs arranged in a ring or ball configuration. These include the PanoCam3d, 360RIZE 3DPRO, Zphereo Z6X3D, and Vuze. However, as discussed in [Anderson et al. 2016], multiple stereo pairs provide an inferior imaging geometry to evenly distributing cameras facing outward in a ring or ball of overlapping field of view cameras, as in the Google Jump and Facebook Surround 360 cameras.

In this paper, we show that even with the smallest number of fisheye lenses (4 for our pair of Ricoh Theta S cameras, as opposed to 6 for the Z6X3D or 8 for the Vuze), we can achieve good-quality spherical stereoscopic capture and display. Furthermore, these other system use proprietary stitching software and/or fail to correctly handle parallax between their stereo camera pairs. The algorithms we develop in the paper for adjusting vertical and horizontal parallax while seamlessly blending over transitions could easily be extended to these alternative camera rigs, which is another contribution of this paper.

2.3 Viewing 360 content

Both YouTube and Facebook now support the display and interaction with 360° content. While monoscopic panoramas can be interactively viewed on mobile devices and Web clients, viewing stereo content requires a head-mounted display of some kind. At the consumer level, Google Cardboard, Samsung Gear VR, and Google Daydream use a smartphone as the display device, which is then imaged through the two lenses to enable stereo viewing. At the moment, the simplest way to find and view content is to use the service provided by YouTube and to view in Google Cardboard or the Samsung Gear VR. We leverage this to demonstrate our system.

3 OVERVIEW

Our capture hardware consists of two Ricoh Theta S 360 cameras mounted on an aluminum bar with holes spaced at standard interocular distances (64mm) as shown in Figure 1a. A central threaded hole accepts all standard tripods. Each camera has two fisheye lenses, each imaging a (greater than) 180° field-of-view. The two images are stitched together by Ricoh software into a 5376×2688 pixel *equirectangular* image for stills. This software compensates for the small missing area between the lenses by warping the images to close the gap. This distortion is addressed by our method. Video is recorded at 30 fps at 1920×1080 resolution, which is on the low end for an immersive VR experience, but the methods presented in this work should generalize to new, higher resolution devices now starting to reach the market. We therefore use currently available devices to validate the concept. Finally, we can view the final content on a number of platforms including the Oculus Rift, Samsung Gear VR, or Google Cardboard, all of which have significant resolution limitations as of this writing.

Two 360° images are obtained from the side-by-side cameras as equirectangular images, 360° across and 180° from bottom to top. We perform a series of steps to transform these original images to overcome most of the problems outlined in the introduction. These include:

- Geometrically align the two images by finding rotational corrections for both cameras such that their orientations are identical and their mutual epipoles (direction of translation) are along the y-axis.
- Photometrically align the two images to adjust for exposure and/or gain differences.
- Compute dense correspondence between the two images to:
  - remove off-horizon vertical parallax; and
  - serve as a basis for equalizing horizontal disparity due to varying inter-occular distance.
- Swap regions of the left and right images to:
  - account for the reversal of eyes when looking backwards; and
  - hide the opposing cameras seen in each other's view.

Each step is detailed in the following sections. We describe each step as resulting in new images, but, in fact, steps are combined into a single warp field to avoid excessive blurring and aliasing due to multiple resampling of the images.

4 GEOMETRIC AND PHOTOMETRIC ALIGNMENT

We begin by finding and applying a geometric and then a photometric transformation to each input image to align them. These aligned images are then used to compute dense correspondence and then stitched together into the omni-directional stereo panorama.

4.1 Geometric Alignment

Because of slight misalignment in the mounts or in the cameras themselves, we cannot expect the stereo pair of images to be oriented in exactly the same direction. Even small misalignments, particularly vertical misalignments, make stereo viewing difficult. Thus, our first step is to find rotations to apply to both images to place them in a parallel coordinate system. They then differ only by the translation that separates the two cameras, which we constrain to lie along the y-axis.

To achieve this, we first remap the equirectangular images into cube maps to minimize local distortions. We then use the Oriented FAST and Rotated BRIEF (ORB) feature detector and descriptor [Rublee et al. 2011] in both images and match the extracted feature points using a brute-force matcher with a cross-check criterion. Using RANSAC, we determine the transformation (rotation + translation) that best explains the largest number matching pairs and then refine the pose using a non-linear optimization applied to the inliers. We then find the pair of smallest rotations that simultaneously align the camera orientations while constraining the direction of translation to be along the y-axis, resulting in the aligned pair (L', R') (Figure 3). We find that, in practice, it is beneficial to iterate on this refinement procedure to alleviate any difference in distortion introduced by the cubemap projection until the number of inliers found by RANSAC stops increasing. In practice, this procedure terminates after 3-4 iterations.

We also found that, in practice, manufacturer software for stitching 360 photos and videos often dynamically warps the imagery near the seams to cover the gap between the two lenses. This prevents RANSAC from quickly finding a single mode of inliers. We address this by blocking out a band of ±20° around the seams and discarding any ORB features detected in this region. This has the effect that the fronts and backs of the two 360 cameras will be well-aligned, but

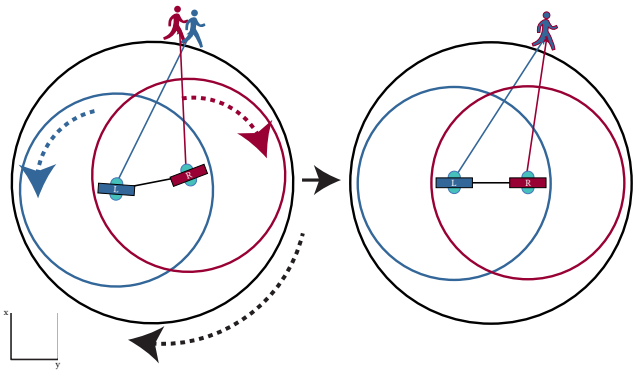

Fig. 3. We rotate both images to align each eye's coordinate system while ensuring that the translations are along the y-axis.

the edges will be less so. We will come back to address the impact of this design decision in later sections.

4.2 Photometric Alignment

Due to small exposure differences or unequal camera responses, the two cameras provide somewhat different values for the same scene points. This is in addition to the different exposures between the front and back cameras on the device which create issues, but also expand the dynamic range when there are very different scene luminances at the front and back. Goldman [2010] offers a solution to the vignette and exposure correction problem. However, the photometric misalignment present in the stitched equirects is not strictly a byproduct of the optics of the device and instead a byproduct of the stitcher's algorithm performing its own compensation. Therefore, we develop a simple technique for our scenario. To photometrically align the two geometrically aligned images, I_1 and I_2, we follow a simple procedure. We blur each image with a large (6% of the image width, e.g., 256 pixels for a 4k equirect) low-pass filter to produce $I_{1,\text{blur}}$ and $I_{2,\text{blur}}$. This avoids the effects of disparity and any remaining geometric misalignments. Then we photometrically align each image to the other with the following per-pixel relation:

$$I_{1\to 2} = \frac{\text{gray}(I_{2,\text{blur}})}{\text{gray}(I_{1,\text{blur}})} I_1 \quad (1)$$

$$I_{2\to 1} = \frac{\text{gray}(I_{1,\text{blur}})}{\text{gray}(I_{2,\text{blur}})} I_2 \quad (2)$$

A simple way to find a photometrically aligned pair *between* these two images would be to take the average, that is $I_1' = (I_1 + I_{1\to 2})/2$ and similarly $I_2' = (I_2 + I_{2\to 1})/2$. However, we found that since the other camera is visible in each image, pixels in the overlapping regions should not be corrected (i.e., We do not want the color of the camera to affect the unoccluded pixels in the other view). We accomplish this by using a sine-weighted blending term between I_1, I_2 and $I_{1\to 2}, I_{2\to 1}$.

$$I_1' = \frac{\sin(\theta)+1}{2} I_{1\to 2} + \left(1 - \frac{\sin(\theta)+1}{2}\right) I_1 \quad (3)$$

$$I_2' = \frac{\sin(-\theta)+1}{2} I_{2\to 1} + \left(1 - \frac{\sin(-\theta)+1}{2}\right) I_2 \quad (4)$$

where camera 2 is visible in I_1 at $\theta = \frac{\pi}{2}$ and camera 1 is visible in I_2 at $\theta = \frac{3\pi}{2}$.

We experimented with modifying the color per-channel, but found that disparity-induced misalignment led to more noticeable color bleeding than if we used a single ratio per pixel. We also experimented with modifying only the exposure of a single image, but this let to asymmetric artifacts between the two eyes.

5 EXPECTED DISPARITY

Next, we examine the remaining vertical and horizontal disparity between the two images after applying the geometric and photometric alignment (Figure 4). The disparity between the two images is due to the offset between the two cameras, with the disparity inversely proportional to the distance to the scene point and proportional to the *apparent* inter-occular distance between the two camera. We say *apparent* inter-occular distance since this changes with the cosine of the horizontal *azimuth* (longitude) angle, θ, away from the normal to the line between the cameras as shown in Figure 6. The apparent distance also varies with the vertical *altitude* angle, ϕ, above or below the horizon (Figure 5).

While horizontal disparity provides the desired depth cues through the vergence of our binocular system, vertical disparity is particularly difficult for our eyes to accommodate, since there is little ability for one eye to gaze higher than the other. We expect to observe very little vertical disparity near the horizon and also in the direction the cameras are facing. But, away from the forward (and backward) direction and closer to the poles, the vertical disparity grows (see Figure 4 (c)). This causes dizziness and eye strain in heads-up viewing if one turns one's head away from a forward direction and looks downward at some angle.

Figure 5 shows the imaging geometry for points lying on a circle on the ground plane. The point directly in front of the pair of cameras will have no vertical disparity since $\phi_{L0} = \phi_{R0}$. However, this is not the case for the point on the circle at $\theta = 90$ lying on the vertical plane through the line connecting the cameras. At a downward angle of 45°, if the radius r equals the height above the ground, h, (say both are 1.6 meters), the angles ϕ_{L90} and ϕ_{R90} will differ by a bit more than 1°.

At this point, you may wonder why this is a problem in our imaging system, since in the real world we can turn our eyes (not our head) to the side and downward and see clearly without a problem. Our eyes are equipped with muscles to turn the eyes left/right, and up/down, but they also have less-known superior and inferior oblique muscles that perform a slight in-plane rotation specifically for looking down and to the side. In our case, we do not have the ability to satisfy this condition while also satisfying the much more common viewing scenario of turning one's head and gazing downward. We therefore need to remove vertical disparity.

The horizontal disparity is what provides the stereo cues for depth, so we wish to retain this. Furthermore, we would expect the

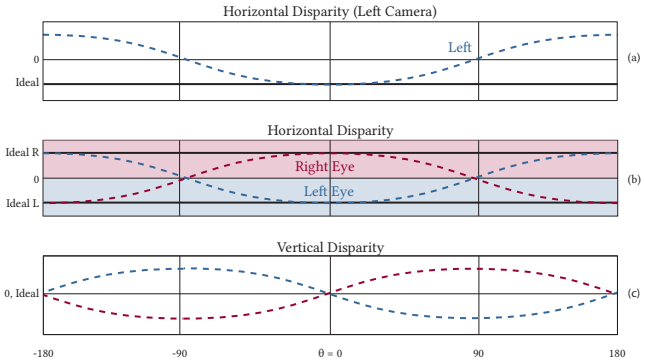

Fig. 4. At a given downward angle from horizontal (i.e., along a scanline in the bottom half of the equirectanglar representation), at a fixed scene depth, we would expect to observe disparities that vary across the image. (a) The horizontal disparities are greatest in the forward direction, zero at 90° and −90° since the cameras are aligned, and are flipped in the backward direction ±180°. These disparities are modulated by actual scene depth. The desired disparity to achieve prefect stereo would be constant independent of direction for a fixed depth. (b) We can get closer to the desired values by simply using the other camera's backward facing hemisphere. Intuitively, the left and right eyes are reversed when we turn around. (c) The vertical disparities are expected to be highest at ±90°. We will cancel out all vertical disparity.

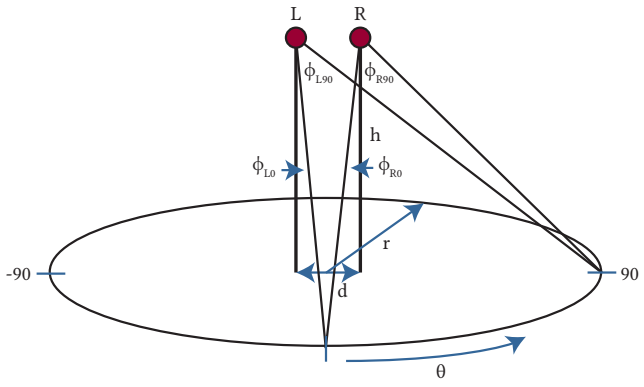

Fig. 5. In the direction the cameras are facing, we expect 0 vertical disparity, however at some altitude below the horizon, as θ moves away from the forward direction, the vertical disparity increases to a maximum at ±90°.

horizontal disparity to stay constant for a fixed depth. Along any scanline (altitude angle) in the equirectangular representation (see Figure 4 (a)), the horizontal disparity is correct only directly in front of the cameras. It falls to zero as one looks left and right 90°, and is in fact exactly reversed if one turns fully around (±180°).

To get closer to the ideal disparities, we can simply swap the back facing portions of each camera to the opposite eye, moving the left camera's back facing section to the right eye and vice versa (see Figures 4 (b) and 8). This gets us closer to the desired disparity and at first glance appears to provide useful 360 stereo. However, the problem of disturbing vertical disparity and attenuated horizontal disparity in the sideways direction remains. The remainder of this

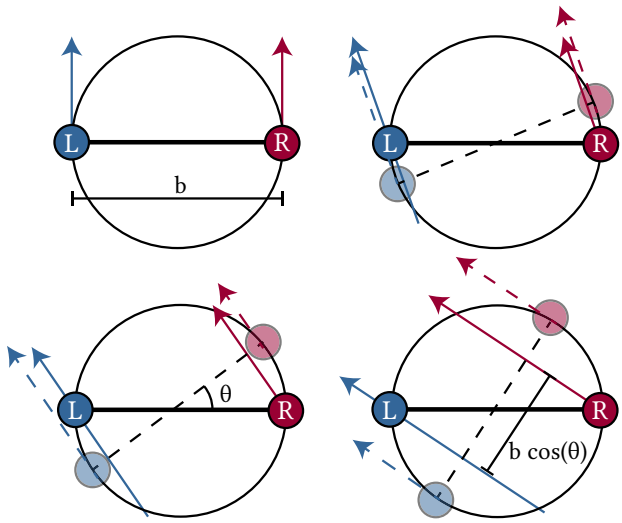

Fig. 6. If rays are naively mapped from the source equirectangular panoramas to the output ODS panorama, as the angle θ deviates from 0, the apparent baseline shrinks by a factor of $\cos\theta$. Our horizontal disparity correction method stretches this apparent baseline back to its correct size by shifting rays to amplify the horizontal disparity measured by our dense correspondence procedure by a factor of $1/\cos\theta$.

section covers how we use estimated dense correspondence to correct these disparities. Section 6 discusses how we estimate the dense correspondence.

5.1 Canceling Vertical Disparity

To cancel out the vertical disparity, we simply use the vertical component of the computed flow maps, divide them in half, and use this in the backwards flow calculation for each image. We have found this step to greatly reduce eyestrain, particularly when looking sideways and down towards the ground; but even areas closer to the horizon, with much smaller vertical disparity, become significantly easier to view.

5.2 Correcting the Horizontal Disparity

Equalizing the horizontal disparity to simulate a consistent disparity for a given scene depth is less straightforward. In Figure 4 (a), we see the expected horizontal disparity, left-to-right, along some scanline for some given depth. The observed disparity is modulated by the inverse depth to the scene. The perceived horizontal disparity is correct only directly in front of the cameras. We can correct the angularly-varying disparity by flowing the image to increase the disparity by a factor of $1/\cos\theta$. In performing our experiments, we have found that the distortion introduced by the manufacturer's stitcher violates epipolar constraints and negative disparity are present in locations other than the back-facing hemispheres of each camera (i.e., near the visible camera in each view, the stitcher often stretches the background behind the camera). If we detect this negative disparity, rather than amplifying it even more with the $1/\cos\theta$ correction, we clamp it to 0, placing the scene at infinity.

5.3 Rendering

We use the forward splatting method of Anderson et al. [2016] to take our source images and render the target ODS panorama considering both the horizontal and vertical disparity correction terms. Briefly, for a pixel in an input image, we compute the target pixel location using the disparity correction terms and then splat four bilinearly-weighted fragments in the target ODS space, storing them in per-pixel fragment lists. Then, for each pixel in the target ODS panorama, we take the list of fragments, sort them according to the length of their displacement vectors and apply the interval-based compositing algorithm of Anderson et al. [2016]. In our implementation, we define the total fragment range constant, k, in terms of disparity in degrees and set it to 2°. Pixels that appear in the ±10° region around the visible camera in each source image (at ±90°, depending on the camera) are discarded and not included in the forward splatting procedure.

Application of this algorithm alone leads to pixels in the target ODS panorama with no contribution, especially near $\theta = \pm 90°$ where the source images are stretched the most in the target panorama. We fill these gaps by generating a foreground warp map by taking, per pixel, the fragment with the largest disparity and storing its source image coordinate in a foreground buffer. We then apply a 5×5 median filter to these image coordinates making sure not to consider pixels without any fragments in the 5×5 window. Pixels with fewer than 20 neighbors in their 5×5 local neighborhoods are invalided in this foreground map. Finally, we linearly interpolate these source image coordinates along horizontal scanlines to fill these gaps.

5.4 Filling the side regions

Two regions still require additional processing. If we examine the regions along the line of sight between the two cameras, i.e., around $\theta = \pm 90°$, we encounter two problems. First, the opposing camera appears in the images. A second problem is that the observed horizontal disparity vanishes, while the $1/\cos\theta$ terms go to infinity to compensate for this. Thus, the simple disparity enhancement defined above becomes unstable in these regions.

We correct for both problems with a simple heuristic. In the region $-110° < \theta < -70°$, both eyes see imagery from the left camera, and from $70° < \theta < 110°$ both eyes see imagery from the right camera (see Figure 8 (a), (b), (c), and (d)). We linearly interpolate disparities across these regions sampling values at the left and right boundaries of each region to seamlessly interpolate between the different frames. For example, to fill region (a) in Figure 8, for each scanline in the final output, we first find the source location of the pixel from the right camera just to the left of the region (Figure 9). We know this already since we have computed the foreground warp mapping as described above. We then use this location in the flow map from right-to-left to find the corresponding location in the left source image, L_0. Similarly, we find the source of the pixel just to the right of the region in the left source image, L_1. We then resample pixels along a line from L_0 to L_1 in the left source image to fill the scanline in the result. This is repeated for each scanline in region (a). Performing a similar process in regions (b), (c), and (d) completes the final images. Note that although both the left and right eyes see pixels pulled from the left source image in regions (a) and (c), they differ due to the opposing disparities. This gives this region an apparent depth interpolated from the neighboring areas.

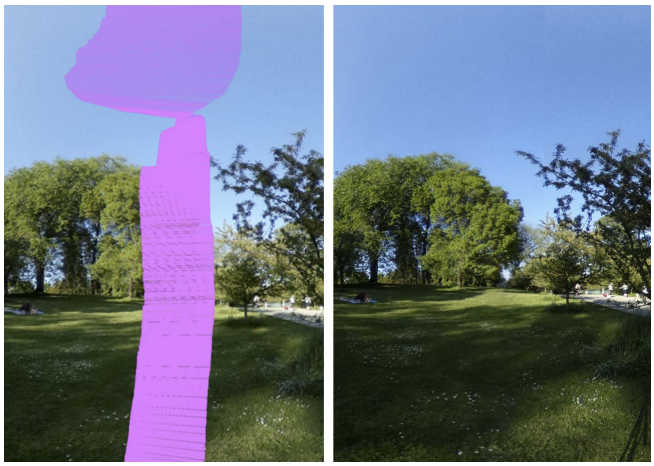

Fig. 7. Left: The pink region in this image corresponds to a rectangular area in the target ODS panorama to be filled in, as in Figure 8 a, b, c, and d. Right: The resulting image with filled rectangular region. Dense correspondence methods such as stereo and optical flow tend to fail in textureless regions, yet the final rendered result often does not exhibit noticeable artifacts precisely because they are textureless, such as is the case with the sky in this example.

In a completely flat scene, every scene point would have a constant disparity from one camera to the other and stitching seams would require replacing an rectangular area with a similar rectangular area from the other image. However, because depth modulates and varies from scene to scene, the area corresponding to rectangle in one image will have a more complex border in the other (see Figure 7). This interpolated disparity works well, but sometimes induces an artifact when there are large depth discontinuities near the region boundaries.

Finally, we perform some additional linear feathering to hide the seams between the two halves of the ODS panorama. A 4° band on the edge of the side region is blended with fragments selected from the alternative camera using the dense correspondence map and a linear ramp for blending.

6 DENSE CORRESPONDENCE-BASED REFINEMENT

In this section, we present two different correspondence estimation algorithms. Section 6.1 describes the optical flow algorithm we apply directly to the equirectangular input images to estimate horizontal and vertical disparity. Section 6.2 describes how to warp the original equirects into a transverse equirect representation to obtain a traditional rectified stereo configuration and how to then post-process the disparities computed with stereo to smooth out the uncertain regions around the epipoles and in low-texture areas.

6.1 Optical Flow

A straightforward approach to computing dense correspondence between two images is to solve the 2D optical flow problem. We do

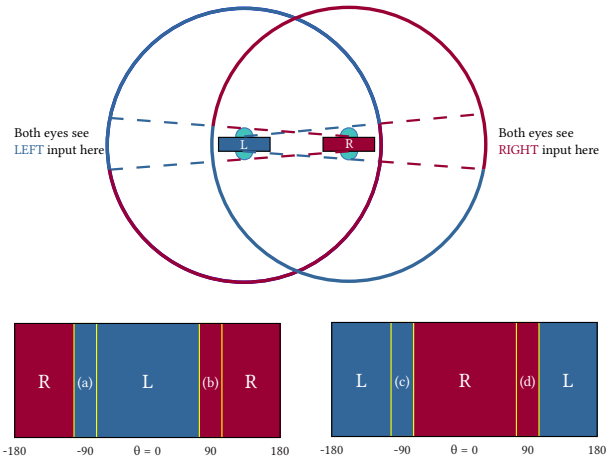

Fig. 8. Each eye is presented an image constructed from one camera in the forward direction, and the opposite camera on the reverse side. In addition, in a small segment to the left and right, both eyes see imagery constructed from the same camera (regions a, b, c, and d). This is to avoid seeing the opposing camera when looking to the side.

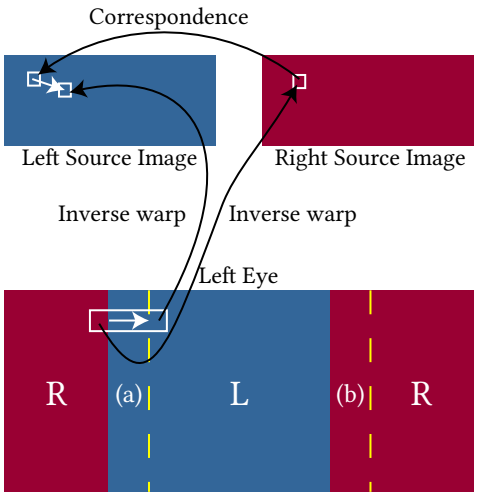

Fig. 9. To fill the regions around θ = -90 and 90, we interpolate disparities. In the example shown for one scanline in region (a), we find the start location in the left source image by inverting the disparity mapping to the right image and then using the flow map from right to left. We then find the ending position through inverse disparity mapping. This establishes a line in the left source from which to pull pixels to fill the scanline for the left eye.

this using two open source optical flow packages, Ce Liu's optical flow [Liu 2009] and EpicFlow [Revaud et al. 2015]. Qualitatively, we found that by skipping the variational energy minimization step of EpicFlow, we were able to get a solution to the flow problem with less compute time and without much degradation in rendering

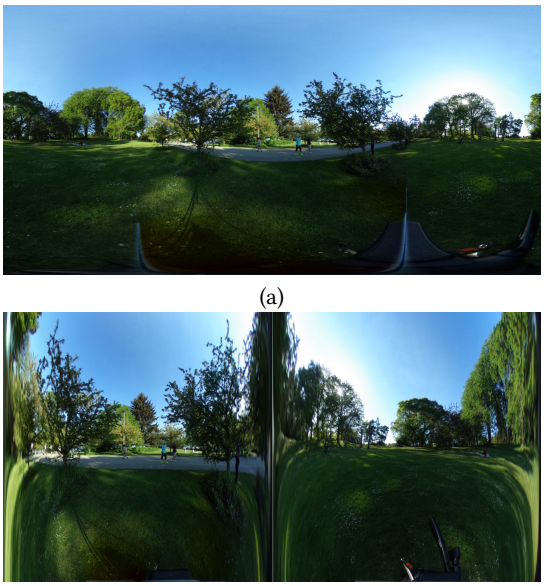

Fig. 10. In order to preserve epipolar constraints, our stereo matcher remaps (a) an equirectangular image into (b) a *transverse equirectangular* projection.

quality for our task. A more detailed comparison is presented in Section 7.

6.2 Rectified stereo matching

Regular optical flow algorithms ignore the known constraints on flow direction implicit in the epipolar geometry (rigid scene assumption). An alternative to computing general optical flow is to first rectify the two images so that horizontal scanlines correspond to epipolar curves and to then apply a classic two-view stereo correspondence algorithm [Scharstein and Szeliski 2002].

To find corresponding epipolar curves in the two input equirectangular images, we observe that epipolar curves are the intersections of *epipolar planes* passing through the two camera centers and the imaging surfaces and are therefore great circles passing through the epipoles.

Figure 10b shows how the original equirectangular image looks like after re-mapping to its corresponding transverse projection. After the re-mapping, corresponding scanlines in the left and right images are shifted versions of each other, with the amount of shift (horizontal disparity) dropping to zero at the boundaries and middle. As mentioned before, the horizontal disparity is proportional to the inverse depth modulated by the cosine of the azimuth (longitude) angle θ.

We match the left and right transverse equirect images with two different open-source stereo matching algorithms, namely LIBELAS [Geiger et al. 2010] and OpenCV's implementation of semiglobal matching [Hirschmuller 2008]. More details of our stereo matcher implementation can be found in Appendix A.

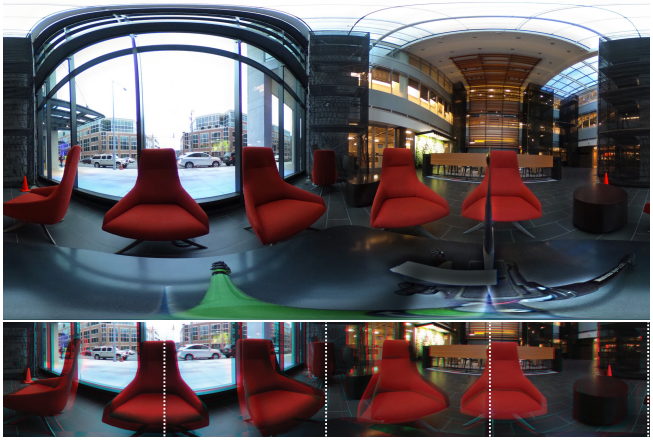

Fig. 11. Top: One image after geometric alignment. Bottom: An anaglyph illustrating the disparity offered by the uncorrected stereo pair. Note the correct disparity to the front (middle of equirect), vanishing stereo to the sides, and reversed stereo to the back (edges of equirect). Dashed vertical lines indicate $\theta = 0, \pi/2, \pi, 3\pi/2$ for clarity.

7 RESULTS

In this section, we present the datasets we acquired, some comparisons between the correspondence algorithms we tested, the design and results of our user study, and the conclusions we draw from these experiments.

7.1 Datasets

In order to evaluate our algorithm, we collected a set of 75 still image pairs and 10 video pairs in a number of settings. Each still image equirect is 5376×2688. Fisheye projection videos are recorded by the device at 1920×1080 and stitched by the manufacturer's software to produce equirects at 1920×960. These video pairs are manually synchronized with a clapperboard, and many have accompanying ambient audio recorded with an additional stereo microphone. This dataset will be publicly released to enable future 360 photography research and will continue to grow over time. Figure 11 shows one such input. Note the anaglyph, which illustrates the stereo effect present in the front of the viewer, the vanishing stereo to the sides, and the swapped stereo to the back.

7.2 Dense Correspondence Evaluation

As noted by others, evaluating image synthesis algorithms that depend on stereo or optical flow as a submodule is challenging, since stereo and optical flow benchmarks often emphasize accurate metric disparity over accurate disparity edges [Barron et al. 2015]. We therefore evaluate the end-to-end quality of the system using a number of means. Several top-performing algorithms with available code from the Middlebury stereo and flow benchmark web site were integrated into our system. To best understand the generality of the system, each stereo or optical flow algorithm we evaluate is set to use the default hyperparameters provided by the authors for the Middlebury benchmark. Since these hyperparameters often focus on metric disparity rather than crisp edges or runtime, we also found

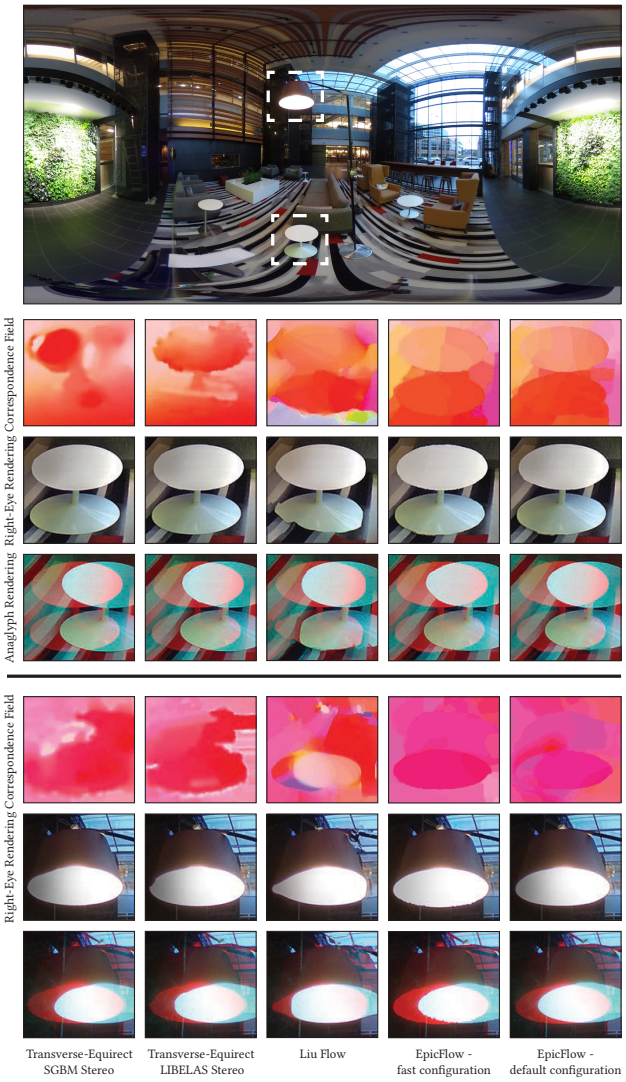

Fig. 12. Top: An input image for the lobby scene, Middle and Bottom: Visualizations comparing the impact of different dense correspondence methods on rendered image artifacts. Different dense correspondence algorithms have errors that produce qualitatively different rendered outputs. Methods such as Semi-Global Block-Matching (SGBM), LIBELAS, and Liu optical flow often introduce warping artifacts such as the ones highlighted on the table and lamp in the image below. EpicFlow emphasizes crisp boundaries and therefore produces a crisp rendering. Given a limited compute budget, EpicFlow can be configured to run much more quickly and retain similar quality at the cost of more jagged edges.

it beneficial to make use of a faster variant of EpicFlow that skips the variational energy minimization step, yet still provides crisp edges and produces quality renderings. This speedup is useful when processing many frames of video.

Figure 12 illustrates the types of artifacts most frequently observed in our rendered panoramas.

Between SGBM and LIBELAS, we found the output of SGBM to be qualitatively better. LIBELAS fills in textureless regions of the scene by making a piecewise planar assumption. Since the transverse-equirect projection is not a perspective projection, many of the assumptions present in the LIBELAS implementation are violated, namely straight lines are no longer straight. SGBM with the thin-plate spline solver to fill in gaps produced a reasonable alternative, since no assumptions about scene geometry are made.

Between Liu's Flow and EpicFlow, we found the crisp flow boundaries characteristic to EpicFlow much more desirable than the smooth flow produced by Liu's Flow. In addition, the quality of EpicFlow did not degrade significantly when we disabled the final variational energy minimization stage, which has the function of smoothing and cleaning up the initial flow solution. This offers a significant speedup useful for processing video.

Both SGBM and LIBELAS suffer from inaccuracies that do not properly align crisply with image boundaries creating "wobbly" image artifacts, which may or may not be consistent between both eyes. While transverse-equirect stereo is a more principled approach compared to solving the general 2D optical flow problem, it suffers from a number of issues introduced by the manufacturer's fisheye image stitcher. In particular, the pixels near the epipoles in both cameras are highly distorted and occasionally produce negative disparity in the aligned, but disparity-uncorrected images (since the stitcher is "stretching" content over that gap). A better approach would be to calibrate the fisheye lenses ourselves and to not use the manufacturer's stitching software. However, this limits the applicability of our method, as we wish to enable novice users who may not wish to calibrate their cameras to produce stereoscopic content. Therefore, we favor optical flow-based methods for the remainder of our evaluations.

All methods except for EpicFlow scale to full 4096×2048 resolution processing. The EpicFlow method is a collection of subsystems (one being the core EpicFlow contribution). In our implementation, we run the Structured Edge Detection [Dollár and Zitnick 2013] stage of EpicFlow at full resolution, but run the DeepMatching [Weinzaepfel et al. 2013] core EpicFlow submodule and variational energy minimization stages at 2048 × 1024 since DeepMatching uses too much memory at 4096 × 2048. A reasonable solution would be to tile the computation rather than keeping all results in memory. Timings are dependent on the particular input. For the `lobby` scene and at 2048 × 1024, transverse-equirect LIBELAS stereo took 116 seconds, transverse-equirect SGBM stereo took 106 seconds, EpicFlow full method took 82.9 seconds, EpicFlow without variational energy minimization took 51.0 seconds, and Liu's Flow took 243 seconds. Timings include both the time to compute the flow from the left camera to the right camera and from the right camera to the left camera. The majority of the time in the transverse-equirect stereo methods was spent in the thin-plate solver, which is dependent on a number of factors such as the density of valid disparity estimates from the initial stereo matching process. It is likely that with a proper fisheye calibration, these initial disparity estimates would be more dense and less time would be spent in the solver filling in the gaps. Timings were measured on a server with 2 Xeon 2.2Ghz 8-core processors. Most public code for these methods were already optimized with OpenMP, SIMD instructions, or both.

Fig. 13. The full-disparity correction variants of the scenes used in the user study rendered as anaglyphs. Top: `troll` scene, Bottom: `lobby` scene.

7.3 User Study

To better understand the impact of the horizontal and vertical disparity correction offered by our approach, we conducted a user study. Participants were sourced by placing a general call at the authors' institution for people in both technical and non-technical occupations and with or without prior VR experience. People with a known history of stereopsis impairment were excluded from the study. The study had 11 participants and each participant completed a pre-study survey form.

Sessions were 30 minutes long and divided into a series of A/B comparisons presented in an Oculus Rift HMD. In each comparison, we selected one of the two scenes from Figure 13. These two scenes were chosen since they offer full, spherical content, cover indoor and outdoor locations, and illustrate the effect of moving subjects, as is the case in `troll`, where the people and car move slightly between the two exposures. For each scene, we applied our algorithm with or without vertical disparity correction and with or without horizontal disparity correction, for a total of four configurations. The EpicFlow without variational energy minimization method was selected as the dense correspondence estimator since it offered the crispest boundaries in the rendered panoramas. In each comparison, two of these configurations were selected at random and designated Option A and Option B (with random naming). Participants were instructed to first explore the scene and provide free-form feedback to the proctor regarding what was good or bad about the scene (what constituted good or bad was left to the participant to decide). After this feedback was collected for both options, the proctor flipped back and forth between the two options and the participant provided a judgment as to which was better in a paired comparison. Each participant was shown as many comparisons as possible in 30 minutes with each shown a minimum of two.

The two scenes used in the study are shown in Figure 13 and are referred to as `troll` and `lobby`. Table 1 shows the paired responses

Condition A Vert.	Condition A Horiz.	Condition B Vert.	Condition B Horiz.	Preferred A	Preferred B
−	−	−	+	6	3
+	−	+	+	6	2
−	+	+	+	2	9
−	−	+	−	1	4
−	+	+	−	2	8
−	−	+	+	2	4

Table 1. Preference response rates for direct comparison between pairs of disparity correction options. Cyan rows indicate comparisons where only horizontal disparity correction was changed, yellow rows indicate comparisons where only vertical disparity correction changed, and magenta rows indicate comparisons where both horizontal correction and vertical correction changed. The actual names for each condition where randomized when presented to the user and renamed for aggregation here.

aggregated under various conditions. When directly comparing panoramas with vertical disparity correction enabled versus vertical disparity correction disabled, participants preferred vertical correction enabled (two-sided sign test, $p = 0.021$). In free-form responses, participants indicated that the vertical disparity uncorrected versions where "blurry", "caused double-vision", or "had uncertain depth". When directly comparing panoramas with horizontal disparity correction enabled versus horizontal disparity correction disabled, there was no clear preference (two-sided sign test, $p = 0.143$). The free-form responses indicated that since the content that ended up in the seam regions differed between the two options, participants were unsure whether to favor different rendering artifacts or improved depth perception. Positive responses for the horizontal disparity corrected versions included "something seems different, but can't tell what; seems closer", "coming at them more in terms of depth", "much more depth", and "world doesn't seem to drop off a ledge anymore" (for `troll`). Negative responses for the horizontal disparity corrected version included "beams don't connect as well" (for `troll`), "legs don't line up as well with body" (for `troll`), "jagged edges on tables" (for `lobby`), and "ghosting on table" (for `lobby`). Similarly, when flipping both correction options, there was no clear preference (two-sided sign test, $p = 0.076$).

7.4 Stills and Video

We present several examples of anaglyph stills in Figure 14. Additional results are shown in Appendix B. More stills are available for viewing with an HMD from our project website.

Applying our algorithm naively to frames of video independently produces some amount of temporal flickering, since none of our dense correspondence methods explicitly account for temporal consistency. This is especially noticeable in videos with a static camera. Instead, we damp out the update rate for the estimated correspondence fields depending on the pixel coordinate of the rendered panorama. For example, content near $\theta = 0, \phi = 0$ will not move much, even with disparity correction enabled, whereas content near $\theta = \pi/2$ might experience large displacements even with small changes in disparity. The damped horizontal disparity for frame

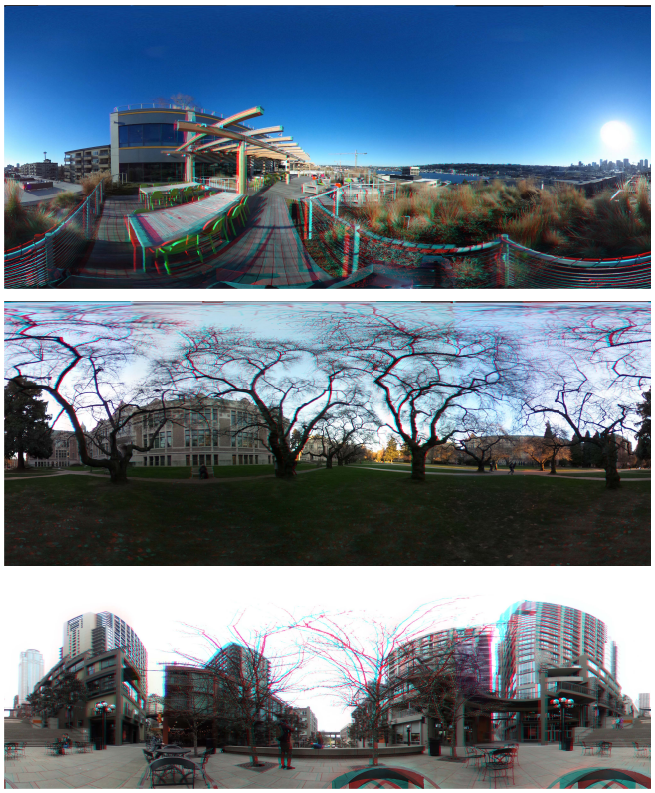

Fig. 14. Examples of stills produced by our system rendered as anaglyphs. The tripod and other supporting surfaces have been blanked out on the bottom.

$i + 1$ at (θ, ϕ), $dx'_{i+1}(\theta, \phi)$, is

$$dx'_{i+1}(\theta, \phi) = (1 - |\cos(\theta)|)dx'_i(\theta, \phi) + |\cos(\theta)|dx_{i+1}(\theta, \phi) \quad (5)$$

and the damped vertical disparity for frame $i+1$ at (θ, ϕ), $dy'_{i+1}(\theta, \phi)$ is

$$dy'_{i+1}(\theta, \phi) = (1 - \cos(\phi))dy'_i(\theta, \phi) + \cos(\phi)dy_{i+1}(\theta, \phi), \quad (6)$$

where $dx_{i+1}(\theta, \phi)$ and $dy_{i+1}(\theta, \phi)$ are the temporally independent horizontal and vertical disparity estimates at time $i+1$. Video outputs are available for viewing with an HMD from our project website.

7.5 Limitations

The free-form feedback from our user study allowed us to better understand which artifacts are the most important to address in our ODS panoramas. In particular, the jagged edges on tables in `lobby` distracted users. These jagged edges were a result of running our fast configuration of EpicFlow. Figure 12 shows that running the full method eliminates those jagged edges. Therefore, while the fast configuration might be suitable for video in terms of runtime, we recommend using the full method for high resolution stills, especially for settings when users will spend several minutes examining the photo.

We also learned that tight synchronization between the cameras is necessary for a good viewing experience. Users were confused

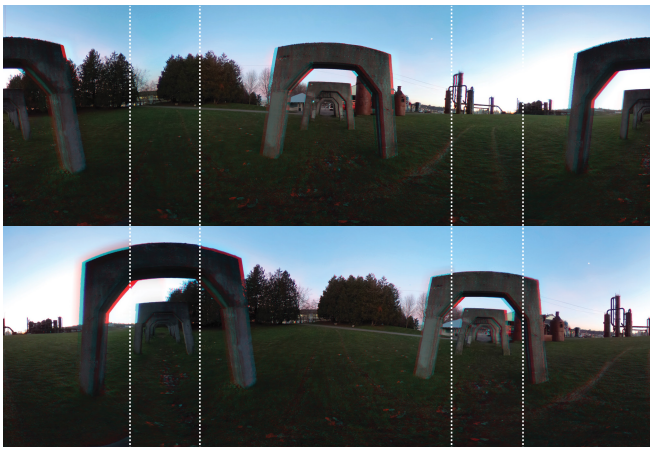

Fig. 15. Two anaglyphs of ODS panoramas taken at similar locations, but with two rig orientations. Top: The baseline of the rig is aligned away from the arches, Bottom: The baseline of the rig is aligned with the arches. Dashed lines indicate the linearly interpolated seam regions near $\theta = \pm 90^\circ$. When authoring content in scenes with vastly different depths, it is advantageous to align the rig with the baseline pointing to far away content in order to mitigate dense correspondence errors and thus stitching artifacts.

by the depth of the car in `troll`, which moves slightly between the exposures of the two cameras. This same artifact affects systems such as the Google Cardboard Camera. This is a need that hardware manufacturers will likely address in the future.

Users were not informed that they were being shown a stitched panorama, but several were able to identify something wrong with the seams of `lobby`. In particular, horizontal wooden beams on one side didn't line up quite right and left a discontinuity and on the other side; some scaffolding also appeared "wavy" across the side-region. A similar phenomenon was observed with the legs of some people in `troll`. These examples illustrate that precise dense correspondence is necessary for convincing stitching. They suggest that a hybrid approach, where cheap correspondence is computed in most places and expensive correspondence is computed near the side-regions, would be beneficial.

Finally, we note that for handheld video sequences, shaking cameras pose a significant barrier to a pleasant viewing experience; this is compounded by a lack of sub-frame synchronization and rolling shutter effects. A solution to 360 stabilization was recently proposed in Kopf [2016]. While it's not clear whether to extend the method to stabilize the pair of rigidly-attached cameras or to stabilize the rendered ODS panorama, it is clear that some stabilization is necessary.

7.6 Content Authoring Recommendations

Our method is designed to make use of the high quality stereo in front of and behind the stereo pair while altering the disparity on the sides to introduce additional horizontal disparity and mitigate vertical disparity. Given these design choices, it's preferable to place interesting nearby objects in front and behind the cameras. Figure 15 shows a scene where there exists interesting, nearby content on two sides of the rig, but very far content on the other two sides. It illustrates how the final stereo panorama degrades when the baseline is aligned *with* the nearby content rather than *against*. Inaccuracies in the dense correspondence field produces more noticeable stitching artifacts when the seam of the panorama is allowed to overlap the nearby content.

8 DISCUSSION AND FUTURE WORK

We have demonstrated that two low-cost $360^\circ \times 180^\circ$ cameras can capture compelling imagery for stereo viewing in a head-mounted display, despite the fact that the effective inter-occular distance varies with viewing angle. We achieve this through rotational alignment, color adjustment, and most importantly, by identifying and adjusting the vertical and horizontal disparities. We do not require the user do offline calibration of their 360 cameras and instead designed our method to robustly handle content from manufacturer-stitched imagery despite image distortions.

Our most surprising finding is that even if the amount of horizontal disparity relative to depth varies somewhat, our perception seems to equalize the effects. This allows us to do less disparity adjustment, and thus create fewer artifacts, and yet still achieve convincing results. It is also notable how effective the vertical disparity removal is in areas away from the horizon.

Our claims have been validated by a user study. An interesting application of this insight would be to do disparity editing, that is, to artificially emphasize or de-emphasize some object in the scene, yet provide a realistic experience for the user [Koppal et al. 2011]. This could aid directors in helping guide the audience through their VR experience.

The methods discussed here should be equally applicable to any of the recent low-cost spherical cameras now reaching the market. Many of these devices, unfortunately, make use of proprietary algorithms and so it is difficult to speculate on their performance. Initial demo content from companies such as Vuze suggest that in many scenarios, high quality stereo can be obtained in a fixed number of directions (e.g., four via stereo pairs), but they also sometimes exhibit stitching artifacts between the input pairs.

As new VR displays become available and as capture devices increase in resolution and decrease in price, we expect to see an increased desire to create stereo content. The algorithms developed in this paper will provide useful tools for correcting, stitching, and enhancing the stereoscopic content produced with such devices and thus accelerate the proliferation of user-generated stereo content in the future.

REFERENCES

Rajat Aggarwal, Amrisha Vohra, and Anoop M Namboodiri. 2016. Panoramic stereo videos with a single camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3755–3763.

Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernandez, Sameer Agarwal, and Steven M. Seitz. 2016. Jump: Virtual Reality Video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)* 35, 6 (November 2016).

Jonathan T. Barron, Andrew Adams, YiChang Shih, and Carlos Hernández. 2015. Fast Bilateral-Space Stereo for Synthetic Defocus. *CVPR* (2015).

Paul Debevec, Greg Downing, Mark Bolas, Hsuen-Yueh Peng, and Jules Urbach. 2015. Spherical light field environment capture for virtual reality using a motorized pan/tilt head and offset camera. In *ACM SIGGRAPH 2015 Posters*. ACM, 30.

Piotr Dollár and C. Lawrence Zitnick. 2013. Structured Forests for Fast Edge Detection. In *ICCV*.

Facebook. 2016. Facebook Surround 360. https://facebook360.fb.com/facebook-surround-360/. (2016). Accessed: 2016-12-26.

Andreas Geiger, Martin Roser, and Raquel Urtasun. 2010. Efficient large-scale stereo matching. In *Asian conference on computer vision*. Springer, 25–38. http://www.cvlibs.net/software/libelas/.

Dan B Goldman. 2010. Vignette and exposure calibration and compensation. *IEEE transactions on pattern analysis and machine intelligence* 32, 12 (dec 2010), 2276–88. https://doi.org/10.1109/TPAMI.2010.55

Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 43–54.

Christopher Grayson. 2016. 3D Cameras and Virtual Reality. http://www.giganti.co/3D-VR-Cameras. (2016). Accessed: 2017-01-16.

Heiko Hirschmuller. 2008. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence* 30, 2 (2008), 328–341. http://docs.opencv.org/java/3.0.0/org/opencv/calib3d/StereoSGBM.html.

Hiroshi Ishiguro, Masashi Yamamoto, and Saburo Tsuji. 1990. Omni-directional stereo for making global map. In *Third International Conference on Computer Vision*. IEEE, 540–547.

Hansung Kim and Adrian Hilton. 2013. 3D scene reconstruction from multiple spherical stereo pairs. *International journal of computer vision* 104, 1 (2013), 94–116.

Johannes Kopf. 2016. 360° Video Stabilization. *ACM Trans. Graph.* 35, 6, Article 195 (Nov. 2016), 9 pages. https://doi.org/10.1145/2980179.2982405

Johannes Kopf, Matt Uyttendaele, Oliver Deussen, and Michael F. Cohen. 2007. Capturing and Viewing Gigapixel Images. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)* 26, 3 (2007), to appear.

Sanjeev J Koppal, C Lawrence Zitnick, Michael Cohen, Sing Bing Kang, Bryan Ressler, Alex Colburn, and others. 2011. A viewer-centric editor for 3D movies. *IEEE Computer Graphics and Applications* 31, 1 (2011), 20–35.

Jungjin Lee, Bumki Kim, Kyehyun Kim, Younghui Kim, and Junyong Noh. 2016. Rich360: optimized spherical representation from structured panoramic camera arrays. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 63.

Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 31–42.

Ce Liu. 2009. *Beyond Pixels: Exploring New Representations and Applications for Motion Analysis*. Ph.D. Dissertation. Massachusetts Institute of Technology.

S.K. Nayar. 1997. Catadioptric Omnidirectional Camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 482–488.

Nokia. 2016. Nokia OZO. https://ozo.nokia.com/. (2016). Accessed: 2017-01-16.

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

F Perazzi, A Sorkine-Hornung, H Zimmer, P Kaufmann, O Wang, S Watson, and M Gross. 2015. Panoramic video from unstructured camera arrays. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 57–68.

Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. 2015. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Computer Vision and Pattern Recognition*.

C. Richardt, Y. Pritch, H. Zimmer, and A. Sorkine-Hornung. 2013. Megastereo: Constructing High-Resolution Stereo Panoramas. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013)* (2013), 1256–1263.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*. IEEE, 2564–2571.

Daniel Scharstein and Richard Szeliski. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* 47, 1-3 (2002), 7–42.

Heung-Yeung Shum and Li-Wei He. 1999. Rendering with Concentric Mosaics. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 299–306. https://doi.org/10.1145/311535.311573

Richard Szeliski and Heung-Yeung Shum. 1997. Creating full view panoramic image mosaics and environment maps. In *Computer Graphics (SIGGRAPH'97 Proceedings)*. Association for Computing Machinery, Inc., Los Angeles, 251–258. http://research.microsoft.com/apps/pubs/default.aspx?id=75673

Kenji Tanaka and Susumu Tachi. 2005. TORNADO: Omnistereo video imaging with rotating optics. *IEEE transactions on visualization and computer graphics* 11, 6 (2005), 614–625.

Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. 2013. DeepFlow: Large displacement optical flow with deep matching. In *IEEE Intenational Conference on Computer Vision (ICCV)*. Sydney, Australia. http://hal.inria.fr/hal-00873592

Christian Weissig, Oliver Schreer, Peter Eisert, and Peter Kauff. 2012. The ultimate immersive experience: panoramic 3D video acquisition. In *International Conference on Multimedia Modeling*. Springer, 671–681.

Fan Zhang and Feng Liu. 2015. Casual stereoscopic panorama stitching. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2002–2010.