An Integrated 6DoF Video Camera and System Design

ALBERT PARRA POZO, MICHAEL TOKSVIG, TERRY FILIBA SCHRAGER, and JOYCE HSU, Facebook Inc.
UDAY MATHUR, RED Digital Cinema
ALEXANDER SORKINE-HORNUNG, RICK SZELISKI, and BRIAN CABRAL, Facebook Inc.

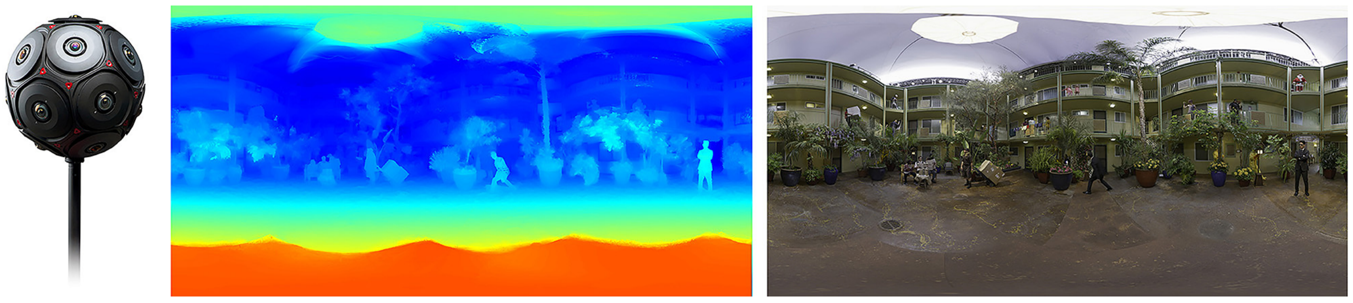


Fig. 1. The commercial 16 camera system, an equirectangular depth map, and final color rendering produced from our system.

Designing a fully integrated 360° video camera supporting 6DoF head motion parallax requires overcoming many technical hurdles, including camera placement, optical design, sensor resolution, system calibration, real-time video capture, depth reconstruction, and real-time novel view synthesis. While there is a large body of work describing various system components, such as multi-view depth estimation, our paper is the first to describe a complete, reproducible system that considers the challenges arising when designing, building, and deploying a full end-to-end 6DoF video camera and playback environment. Our system includes a computational imaging software pipeline supporting online markerless calibration, high-quality reconstruction, and real-time streaming and rendering. Most of our exposition is based on a professional 16-camera configuration, which will be commercially available to film producers. However, our software pipeline is generic and can handle a variety of camera geometries and configurations. The entire calibration and reconstruction software pipeline along with example datasets is open sourced to encourage follow-up research in high-quality 6DoF video reconstruction and rendering [1].

CCS Concepts: • **Computing methodologies → Camera calibration**; **3D imaging**; **Matching**; **Computational photography**; **Virtual reality**; *Reconstruction*; *Image-based rendering*;

Additional Key Words and Phrases: Video Stitching, 6DoF

ACM Reference Format:
Albert Parra Pozo, Michael Toksvig, Terry Filiba Schrager, Joyce Hsu, Uday Mathur, Alexander Sorkine-Hornung, Rick Szeliski, and Brian Cabral. 2019. An Integrated 6DoF Video Camera and System Design. *ACM Trans. Graph.* 38, 6, Article 216 (November 2019), 16 pages. https://doi.org/10.1145/3355089.3356555

[1] Available at https://github.com/facebook/facebook360_dep

Authors' addresses: Albert Parra Pozo, app@fb.com; Michael Toksvig, tox@fb.com; Terry Filiba Schrager, tschrager@fb.com; Joyce Hsu, joycehsu@fb.com, Facebook Inc. Uday Mathur, uday@red.com, RED Digital Cinema; Alexander Sorkine-Hornung, alexander.sorkine-hornung@oculus.com; Rick Szeliski, szeliski@fb.com; Brian Cabral, bkc@fb.com, Facebook Inc.

1 INTRODUCTION

Head motion parallax or full 6 Degrees of Freedom (6DoF) imagery is an essential element of the human visual system [Aytekin and Rucci 2012] and critical for compelling Virtual Reality immersion into the real three dimensional world [Overbeck et al. 2018; Thatte et al. 2016]. 6DoF imagery moves beyond traditional pure stereo and omnistereo [Peleg et al. 2001], providing a more physically accurate and natural immersive experience. However, designing the hardware and software for a video capture system to support even a modest form of full 6DoF head motion parallax involves many technical challenges.

True immersion requires fully spherical capture that does not compromise one part of the spherical environment over another. For instance, horizontally arranged rigs such as those described in [Anderson et al. 2016; Facebook 2016; Schroers et al. 2018] only provide horizontal binocular parallax; even a slight head roll or tilt produces incorrect and disorienting visual results. Full 6DoF head-motion parallax requires having enough cameras arranged to evenly capture the full spherical 4π steradians, while providing sufficient overlap between the individual cameras' field-of-views to extract the necessary scene depth information. Doing so with a minimal, practically feasible camera configuration is a complex design problem.

In this paper, we describe our complete 6DoF 360° video capture and playback system, including our design methodology, camera architecture, and associated computational imaging and rendering pipelines. Our software pipeline is general enough to handle a wide variety of camera arrays.

Our paper's contributions include:

- the design of a full 360° 6DoF, high quality, video hardware and software pipeline
- a novel camera design methodology, specification, and optimization procedure
- a high precision markerless intrinsic and extrinsic calibration procedure

- a multi-view high-overlap stereo depth reconstruction system
- a generalized view synthesis pipeline for arbitrary camera arrays including depth estimation and rendering
- open sourcing of high quality per camera footage from many scenes and the entire software pipeline

Section 2 contains a review of related literature, followed by a high-level overview of system components in Section 3. The next two sections cover the camera design and camera calibration methodology. This is followed by a description in Section 6 of our multi-view depth reconstruction algorithm. Our corresponding real-time image-based rendering algorithm is presented in Section 7. Section 8 presents our results, highlighting our advantages over traditional omni-directional stereo and our ability to composite CGI elements. The final Section 9 concludes with a discussion of opportunities for future work.

2 RELATED WORK

The popularity of 360° films stretches back to the late 1950s with the introduction of Disney's *Circle-Vision* 360° film *America the Beautiful* at the 1958 World's Fair in Brussels [Disney 2008]. Over the last two decades, more compact and affordable capture rigs have been developed (see [Uyttendaele et al. 2004] for a review), culminating in today's hand-held consumer 360° video cameras. A large number of 360° videos can now be found on the Web for viewing in VR headsets [Circle 2018; Disney 2016].

When viewed in Virtual Reality, 360° videos are much more compelling if displayed stereoscopically, i.e., with different left and right eye views. A popular way to achieve this is to encode a separate 360° video or panoramic still image for each eye. This approach is called *Omni-Directional Stereo* (ODS), and was first suggested by [Ishiguro et al. 1990] and [Peleg et al. 2001]. An economical approach to capture an ODS of a still scene is to rotate a video or still camera on an offset rig or to capture an equivalent hand-held video. Combinations of stereo matching and/or optical flow can then be used to create the desired left and right views [Hernandez 2016; Richardt et al. 2013].

To film dynamic omnistereo videos, multiple video cameras can be arranged in a circular ring and then post-processed to produce the left and right eye videos. Examples of such systems include the TiME Lab omni-directional omnistereo multi-camera system [Weissig et al. 2012], Google's Jump virtual reality video system [Anderson et al. 2016], and Facebook's Surround 360 system [Facebook 2016]. It is also possible to use a pair of compact 360° video cameras to generate omnistereo video [Matzen et al. 2017], or to use a rapidly spinning two-camera rig to produce such content [Konrad et al. 2017].

While omni-directional stereo systems are popular, they do not support true head motion parallax and also produce artifacts when the viewer tilts or rolls their head [Anderson et al. 2016; Matzen et al. 2017; Overbeck et al. 2018; Thatte et al. 2016]. Concentric Mosaics [Shum and He 1999] can support limited side-to-side parallax, but require storing a large number of video frames and still produce distortions with forward or upward motion.

To overcome these limitations, some systems use *geometric proxies* or *view interpolation* techniques originally developed for image-based rendering systems. While the earliest lightfield capture and rendering systems such as the Lumigraph [Gortler et al. 1996] and Unstructured Lumigraph [Buehler et al. 2001] used global 3D mesh proxies, more recent systems such as [Hedman et al. 2016; Overbeck et al. 2018; Penner and Zhang 2017; Zitnick et al. 2004] estimate per-camera depth maps or pair-wise flow fields [Bertel et al. 2019] and use these to warp the source images into the desired novel viewpoints. Other approaches focus on adaptive meshing and mesh tracking to create temporally coherent geometry [Collet et al. 2015]. We review the related literature on multi-view stereo reconstruction in Section 6.

To achieve high rendering quality, many of these systems use a large number of images and cameras swept on an circular arc or spherical surface [Bertel et al. 2019; Overbeck et al. 2018], which restricts their use to still scenes. For example, the system built by [Overbeck et al. 2018] takes between 30 seconds and 33 minutes to acquire a 360° scene. The depth-augmented omni-stereo system developed by [Thatte et al. 2016] uses two Ricoh Theta 360° video cameras stacked above each other, and so can capture video, but has view-dependent parallax quality, as does the system developed by [Matzen et al. 2017].

Our system is the first to produce a high-resolution 360° video experience with geometrically correct left and right eye images at interactive rates. Like some of the previous systems, it estimates per-input depth maps; it also de-noises these depth maps temporally to reduce flicker (Section 6). It then uses a discontinuity-aware rendering algorithm to eliminate ghosting artifacts due to stretched mesh triangles (Section 7).

Creating a hemisphere of video cameras for 6DoF limited head-motion parallax is not a new idea [Milliron et al. 2017]. Work by [Afshari et al. 2013] describe a *Panoptic* camera composed of 100 VGA CMOS sensors arranged in a hemispherical grid. Similar in spirit to subsequent lightfield cameras it used small baseline of many VGA video cameras to capture a hemisphere of data. It differs from our approach in that they optimized for angular light field resolution over spatial resolution. As described below, we make a fundamentally different trade-off for greater spatial resolution and lower component count, which increases the overall system reliability and image quality.

3 OVERVIEW

Taking a blank-sheet approach and learnings from several earlier prototype systems, we stepped back and asked what it would take to design a camera taking into account numerous factors, including optimal camera placement, overlap, field of view, large head-box parallax, resolution, ruggedness, thermal design, streaming, calibration, depth reconstruction, and real-time rendering. We designed it to be up and running within 30 minutes[2] from unboxing to recording - something we've tested and achieved. However, meeting all of these competing design parameters comes at a cost. Our rig is 1 meter in diameter and weighs ∼ 37kg and is not portable, but is relatively easy for two grips to handle and move around on set - again something we've actually tested.

[2]Most professional cameras take around 10 minutes to setup.

In this section, we provide a high level overview of key design considerations. In Section 4 we then dive deeper into the technical contributions of this work.

3.1 Camera Placement

An important insight that guided our early design work is that there is no computational reason to axially align the cameras. All previous multi-camera designs have aligned the cameras in a regular arrangement. It is conceptually simpler if you use rectilinear lenses arranged in a regular pattern, e.g., a grid [Levoy and Hanrahan 1996], since epipolar curves are lines and camera-to-camera projections are affine homographies. This approach optimizes for the ease of computation, system reasoning, and algorithm design.

By relaxing this constraint, we instead optimize for the best placement of rays or pixels with the least amount of hardware to accomplish the ultimate goal of the system, namely to compute novel views from captured video frames. At first glance, this seems like it would place a large software burden. But as we demonstrate in Section 6, it is no harder with the appropriate software abstractions to work with arbitrary non-linear camera mappings while incurring only a modest additional computational overhead.

A generalized non-linear camera mapping considerably increases options. Specifically, combining the concept of arbitrary camera placement with fisheye wide angle, f-theta lenses increases overlap and minimizes camera count. Minimizing camera count is not just a cost consideration, but also increases overall system reliability.

3.2 Mechanical Design Considerations

A complete camera design must consider workflow and practical usage scenarios. Since the design was targeted at cinematographers, the camera had to fit into existing on-set environments. We took numerous design steps to ensure our camera met the rigorous demands of on-set workflows, as shown in Figure 2. Our hardware needs to be physically sound and rugged. We conducted finite element simulations to ensure that no modal or resonant frequencies existed that would interfere with the video recording. We studied physical stress knowing that the weight of 16 professional cameras and electronics could warp or bend the cameras and their rigging. For instance, if mounted sideways for hours, the metal chassis could fatigue, bend or sag. The internal spherical core could only be manufactured with state-of-the-art synchronous machining technology to maintain a tolerance of 10 arc-minutes from sensor to sensor.

Physically packing numerous cameras close together and compressing high resolution video generates significant heat. The main thermal challenge is to control each sensor's individual thermal system to maintain optimal temperature within the spherical orientation, where some sensors are in the sun and others are in the shade. Each 8K sensor module used in the 16 camera system needs to dissipate 25 watts. With that in mind, we conducted numerical air flow analyses using commercially available Navier-Stokes airflow software to ensure that the camera could run for hours and not overheat in environments up to 40°C.

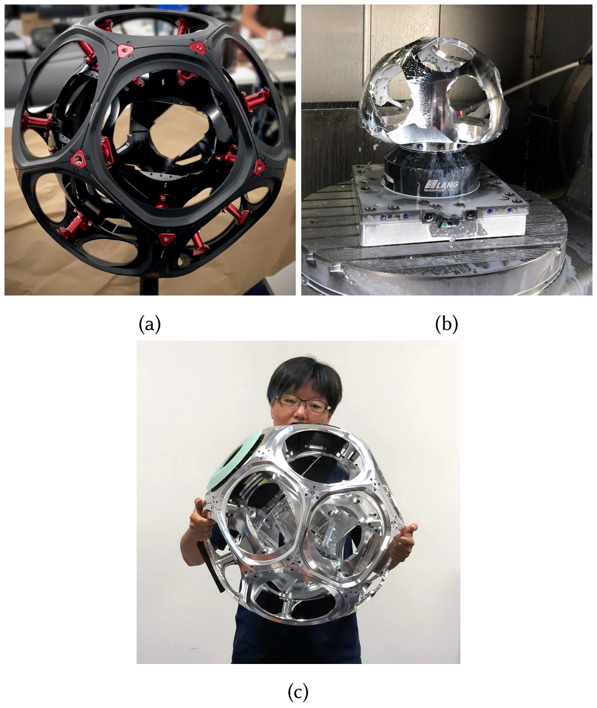


Fig. 2. Double wall structure (a) and machining process (b) illustrates the ruggedness and mechanical precision needed for a professional VR camera. Precision milled skeleton chassis before black anodizing with a person showing the rig's scale (c). The entire rig diameter is 1 meter.

3.3 Optics

Optics are key component of any camera design, especially one where they are a critical design parameter as described in Section 4. Our choice of a 180° f-theta fisheye lens comes from the optimization procedure. Another desirable property to was to maintain sharpness across the full 180° field of view. Most photographic lenses allow for as much 50% center-to-edge MTF[3]. MTF is often traded off against other lens design criteria such as chromatic aberration. However, a camera with such a wide field of view requires extracting as much sharpness as possible. It is a desirable property for the lens and sensor to have some aliasing in order to recover some super resolution upon final image reconstruction. Furthermore, removing the majority of the chromatic aberration is easily accomplished as a per channel warp to align the red and blue channels to the green channel. Since we need to perform a per camera rectification for subsequent depth processing, correcting for radial chromatic aberration comes nearly for free. Additional post-processing would be required to handle other types of chromatic aberration.

Another system design consideration is weight and ruggedness of the optical stack. Keeping the lens under half a kilogram was important considering there are 16 lenses. Also minimizing the mechanical elements in the lens was quite important. This led to a

[3]Modulation Transfer Function, is the common measure of optical sharpness. Both tangential and sagittal MTF fall off [Hecht et al. 2002] are often used describe radial and transverse radial sharpness respectively.

custom Schneider designed F4 fixed focus lens weighing less than 300 grams with a hyperfocal distance of 2 meters.

3.4 Sensor Considerations

The heart of any camera system is the sensor. Based on previous prototypes, it was clear that maximizing the Signal to Noise Ratio (SNR) and thereby increasing dynamic range was critical to both cinematic applications and for resolving depth in high dynamic range scenes. Ideally, a cinematic sensor has a dynamic range of well over 84dB (i.e., 14 bits of dynamic range).

Another important feature of a professional video camera is the ability to shoot at a variety of frame rates and exposure ranges. Specifically in immersive VR applications, too low frame rates may induce visual discomfort. Hence, we designed our system to achieve capture rates of at least 60fps, with an option for 30fps for lower light situations. The creative control of motion blur vs exposure at a given frame rate is controlled by shutter angle[4]. Cinematic professionals want complete control over both shutter angle and frame rate, something that is all the more important in our camera since it has a fixed aperture lens.

Given the wide field of view of our lens, we needed a large high resolution sensor reaching beyond 30 Pixels Per Degree (PPD). The RED Helium sensor meets all of these requirements and is the main reason RED co-designed and built the 16 camera rig. Each sensor has 8,192 horizontal pixels and 4,320 vertical pixels. Since the lens focal length is around 7.5mm, the actual field of view spans 6320 pixels at 180° horizontally and 4,320 pixels at 120° vertically. This produces around 35 PPD, more than double what most VR headsets are capable of displaying. This provides a fair degree of future proofing and motivated our decision for high spatial resolution. The sensor does not have the classic anti-aliasing filter by design so that overlapping images super sample one another. All 16 sensors are clock synchronized and equally exposed.

Rolling shutter vs. global shutter is another design parameter that is important in a multi-camera system. Ideally, all the cameras see exactly the scene at the same time, something that global shutter sensors by definition provide. Historically, global shutter technology comes at a fairly steep trade off against dynamic range. In the end, we opted for greater dynamic range over the benefits of a global shutter.

One concern with rolling shutter is that a fast moving object will appear at different times on different sensors - an inherent problem of arranging rolling shutter cameras on sphere. In order to have good reconstruction, at least two spatially overlapping cameras must also be closely overlapped in time. Interestingly our increased spatial overlap results in a significant improvement for the worse case temporal skew. The formula for this worse case maximum temporal skew is given by the distance between the closest two cameras, in time, for each point on the sphere:

$$\underset{\theta,\phi}{\arg\max}\{\underset{c,c'}{\arg\min}\,\|t_c - t_{c'}\| \forall c, c' \in [C; {}^{C}/_{c}]\} \forall \theta, \phi \in [0, 2\pi; 0, \pi],$$

where C are all cameras visible in the direction of θ, ϕ and t_c is the time from the top of the frame of camera c. Using the above formula we get for the 16 camera arrangement clocked at 60 frames per second a worse case temporal skew of ~2.8ms and an average skew of ~0.5ms. If we take a subject running past the camera at 2 meters this translates to ~9.3 pixels, worse case, and ~1.6 pixels on average.[5] Figure 3 is an equirectangular image of this formula over the sphere for the 16 camera use case. Note that there are very few regions that reach the maximum temporal distance of ~2.8ms.

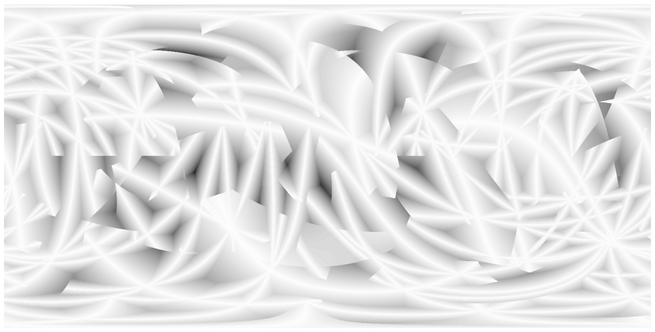


Fig. 3. An equirectangular image showing rolling shutter sweeping patterns formed by the nearest two cameras, in time, across the entire sphere at a distance of 2 meters. White is zero time skew, or perfect temporal overlap, and black represents the maximum of ~2.8ms temporal skew.

While one could imagine simultaneously optimizing the camera arrangement over time and space instead of just space as we describe in Section 4, we decided not perform this optimization given the low probability of worse case failures and very high average temporal overlap; something confirmed in our testing and illustrated in the Supplemental Material video. All-in-all this seemed like a more than acceptable trade-off for choosing a higher dynamic range rolling versus a global shutter sensor.

3.5 Live Monitoring and Data Streaming

Nearly all digital and film cameras support live monitoring feeds so that the on-set creative team (e.g., the Director of Photography) can watch the live action as it appears "on film." This becomes even more crucial for a full 360° camera, since none of the creative staff can stand behind the camera. The only viable solution is to provide a live video feed of the cameras to off-set monitors, ideally ones that can be viewed in full 360°. We do that by providing 4 full frame 4k SDI video outputs from any 4 of the 16 cameras. Additionally, a 4k composite SDI output image of all 16 cameras is provided so that on-set staff can monitor the full camera functionality.

Simultaneous with live previewing of the capture the actual frames must be recorded onto digital media. Compressed raw Bayer data is streamed to a high speed SSD RAID array at approximately 90Gb/sec. This requires custom high bandwidth hardware so that no frames are dropped.

[4]Shutter angle = $360 * \frac{exposure\ time}{frame\ interval}$

[5]Assuming a spatial resolution of 35 pixels per degree and an average human running speed of 12,000 meters per hour, or ~0.0033 meters per ms, our worse case of ~2.8ms temporal skew translates to ~0.0083 meters. So at 2 meters away we get 9.3 pixels $\approx 35 * \frac{180}{\pi} * \arctan\frac{0.00833}{2}$

3.6 System Calibration and Depth Reconstruction

The captured data stream is reconstructed into color and depth images, as described in Section 6. The depth estimation subsystem relies on a camera description file that encodes the camera intrinsic and extrinsic parameters. Each camera configuration has a nominal as-built description file. The marker-less calibration procedure described in Section 5 updates the as-built description and adjusts the parameters of the actual rig. This is critical for high quality post-production work flows since the cameras' actual intrinsic and extrinsic positions will shift with use and handling. The automated calibration procedure also acts like a system check to detect if any camera is wildly out of specification or otherwise dysfunctional.

3.7 Rendering

The purpose of computing depth is to support high-quality novel view synthesis in VR. We use a mesh-based depth compositing technique described in Section 7 that relies on per-camera depth maps and meshes. Each camera's mesh is created from the camera's depth map and textured with that camera's image. The meshes are then rendered from whichever novel viewpoint one desires. This can and does result in a variety of artifacts, discussed below, for which we provide means to ameliorate. We opted for a more classical, geometric depth rendering approach because it is compatible with existing 3D CGI and compositing workflows, making it relatively easy to integrate additional virtual content and visual effects such as relighting and properly occluded 3D objects.

4 CAMERA DESIGN AND OPTIMIZATION

Optimizing the camera arrangement while minimizing hardware complexity raises novel problems of exactly what is optimal and what design and optimization strategy to follow.

The purpose of our rig is to record enough information to allow reconstruction of novel viewpoints within a volume, which we call the *eye box*. We accomplish this by providing a quasi-uniform sparse sampling of viewpoints (camera positions) around that eye box, which informs the first stage of our optimization, namely computing optimal camera positions.

The second stage of our optimization orients each camera to guarantee that every point in the scene is seen by as many cameras as possible. In practice, we use a set of points evenly distributed on a shell of finite radius around the camera rig as a proxy for all the scene points, as described in Section 4.2.

Finally the remaining free variables are optimized to accommodate thermal constraints.

4.1 Camera Positions

A desirable design criterion, based on cost minimization and coverage maximization, of a rig is that it has a uniform inter-camera distance. For example, if one were to arrange the cameras in a horizontal ring, one would space them equally to provide a good sampling of viewpoints in a horizontal plane. This would provide the ability to reconstruct high-quality novel viewpoints within a 2-dimensional disc encompassing the camera ring. As the novel viewpoint moves above or below the plane of this disc, however, the quality of the reconstruction will deteriorate because unseen volumes become visible above or below every horizontal edge. In other words, the eye box is flat.

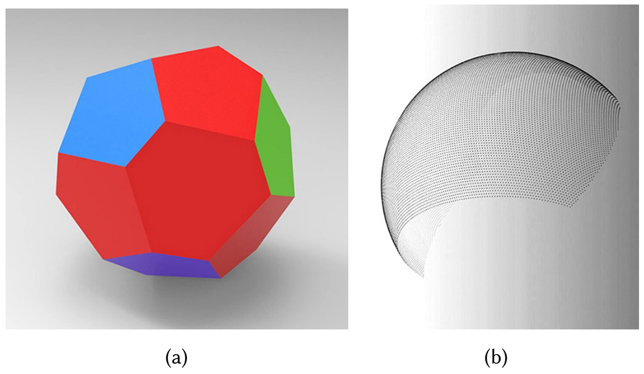


Fig. 4. Camera placement determined by the Thomson simulator for the 16 camera rig (a) and portion of the sphere seen by a single camera (b)

To support novel viewpoints within a volume and not just a plane, the cameras must sample viewpoints that are evenly distributed near the surface of that volume. This leads to our design, which distributes the cameras uniformly across the surface of a sphere.

It is obvious how to uniformly distribute, e.g., 4 cameras on a sphere: Simply place a camera at each vertex of a tetrahedron. Less obvious is what happens when you want to add more cameras. One might think the solution is to place them on the faces or vertices of well known 3D solids, but solids don't exist for all camera counts such that they meet our equidistant placement criterion. In fact, only 4 of the Johnson solids meet the criterion: 4, 6, 12, and 24.

The general problem is one of placing n cameras on a sphere such that they have approximately equal baseline to all nearest neighbors. This is analogous to a well-known physics problem known as the Thomson problem [Weisstein 1998]. It was formulated in 1904 as a means to determine how electrons would distribute themselves on a sphere as they repel each other according to Coulomb's law. A general closed form solution has only been worked out for a handful of cases.

We therefore solve the nonlinear optimization problem

$$\arg\min \sum_{(p,q)\in S} D(p,q) \tag{1}$$

numerically [Ridgway and Cheviakov 2018], with $D(p,q)$ representing the Coulomb force between each pair of camera positions on the sphere S, to find the camera placement that meets our uniform baseline requirement. For this we used Ceres Solver [Agarwal et al. 2012].

This optimization works for arbitrary n, but as we discuss below, the 16 camera arrangement meets all of criteria for overlapping coverage, while at the same time the power of two also provided a nice match to the underlying data path logic needed to aggregate the massive amount of data being generated.

The resulting 16 camera system has a tetrahedral symmetry, with 4 neighborhoods surrounded by 6 cameras, and 12 neighborhoods surrounded by 5 cameras, as shown in Figure 4a.

4.2 Camera View Optimization

Once we have chosen the camera positions to meet our baseline uniformity requirement, we must determine the optimal camera orientation by twisting each camera around its optical axis. Twisting obviously makes no difference if the entire image circle fits on the sensor. As described below, we specify a 180° field of view to optimize for view overlap, but required a large, cropped image circle to preserve sensor resolution. This lens produces an image circle that extends above and below the sensor. A single camera thus has a view of a hemisphere that is missing a segment from the top and the bottom, as shown in Figure 4b.

When the camera sees only part of the image circle, twisting it around the optical axis changes which part of the hemisphere it sees, and thus how it overlaps with other cameras. The total overlap cannot be changed. However, rotating the cameras greatly influences how evenly the overlap is distributed.

A point that is very close to the rig may not be seen by any cameras. Conversely, points farther from the rig are visible to an increasing number of cameras. This leaves us free to select a shell radius at which to optimize the overlap. The overlap deteriorates inside this shell, so the selected shell radius should reflect the nearest points that must be reconstructed with the highest fidelity.

In our approach, we seek to minimize the worst case overlap on that shell, i.e., we seek a set of camera rotations (twists) θ_c that maximizes the overlap in the worst case point on the shell:

$$\arg\max_{\theta_c}\left(\min_{d\in S} o(d)\right), \tag{2}$$

where $o(d)$ is the ordinal-valued overlap function for a point d drawn from the set of all points on the shell S.

This search space is obviously much too large to simply explore exhaustively. Neither the min nor the non-differentiable overlap function lend themselves to directly using a nonlinear solver.

We begin by first softening the overlap function so that it becomes differentiable. Instead of determining whether a point is inside, e.g., the vertical bounds of the sensor by testing the $y \in [-1, 1]$ normalized device coordinate using a step function, we use a soft inside test that uses a sigmoid function instead:

$$h(y) = \frac{1}{1 + e^{-k(|y|-1)}}. \tag{3}$$

Building the overlap function using similar soft inside tests for horizontal and radial values x and r, i.e.,

$$o(d) = \sum_c h(x(d,\theta_c)) \cdot h(y(d,\theta_c)) \cdot h(r(d,\theta_c)) \tag{4}$$

results in an overlap function that is visually similar to the original, yet differentiable (see Figure 5).

The slope at the transition is controlled by the constant k, which we set to $k = 100$. We complete the regularization process by replacing the the infinity norm ℓ^∞ using a finite exponent norm implied by the min function:

$$\min_{d\in S} o(d) = [\max_{d\in S} o(d)^{-1}]^{-1} = \|o(d)^{-1}\|_\infty^{-1} \approx \|o(d)^{-1}\|_p^{-1}. \tag{5}$$

In practice, $p = 20$ works well, providing a sharp enough decision boundary while remaining numerically stable.

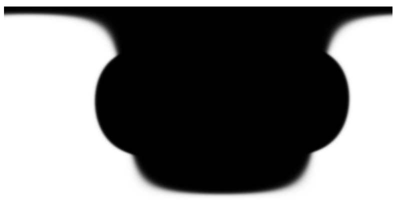


Fig. 5. One of our soft overlap functions shown in equirectangular projection

With the objective in a differentiable form, the problem is solvable using a non-linear solver (we used Ceres Solver) to optimize over the shell. This leaves open the sampling strategy. One could use stochastic gradient descent, which is slow. Uniform sampling introduces structural biases that may align the cameras to the grid. Even using a Fibonacci spiral, which seems to uniformly sample the shell, allowed the fairly straight edges of a camera's view to line up with the lattice-like structure of the spiral. Instead, we use a fast algorithm to produce *green noise* samples, i.e., samples that are approximately uniformly spaced but without any structure. The algorithm simply generates many more samples than it needs, then iteratively rejects the surplus, starting with the points that are most crowded. It is easy for the solver to get stuck in a local minimum, so we solve multiple times using random initial orientations and keep the best solution. Each solution is evaluated by sampling at M points, where $M >> N$. This time the points are taken from a Fibonacci spiral as the lattice structure can no longer torque the solution. We use $N = 3000$ and $M = 300,000$. We define the quality metric to be optimized as

$$\mathit{quality} = \mathit{worst} + \mathit{better\ fraction}, \tag{6}$$

where '*worst*' is the overlap in the worst direction and '*better fraction*' is the fraction of directions where the overlap is better than that. The latter term allows us to give credit to, e.g., a solution that has overlap 5 across 99% of the shell but only 4 in a small spot versus a solution that has overlap 4 everywhere. The quality of the former is 4.99 and the latter is 4.00.

The solutions do not improve significantly beyond a few hundred runs, which complete within an hour on a laptop computer.

4.3 Remaining Free Variables

The preceding sections produce a rig without scale. The overlap will be the same for a rig with a radius of 2 or 200 mm. When manufacturing a physical rig, however, the scale will be constrained by the physical dimensions of the camera modules. We determined the scale manually using a parametric CAD model.

Furthermore, each camera can be rotated 180° without changing the quality of the reconstruction. We exploited this degree of freedom to steer the airflow from each camera module's exhaust clear of the intake for the other camera modules. This optimization was also performed manually using a CAD model.

The focal length and field of view of the lens, as well as the number of cameras, can be thought of as additional free parameters that must be optimized. The focal length determines the size of the image circle on the sensor, which in turn affects both the spatial resolution measured in Pixels Per Degree (PPD) and the amount of

Table 1. Overlap quality

	30 pix/deg	35 pix/deg	40 pix/deg
12 cameras	4.98477	4.40538	3.88276
14 cameras	5.89005	4.99175	4.66438
16 cameras	6.74184	5.92205	4.99726
18 cameras	7.70447	6.82917	5.94449
20 cameras	7.99949	7.65927	6.82848

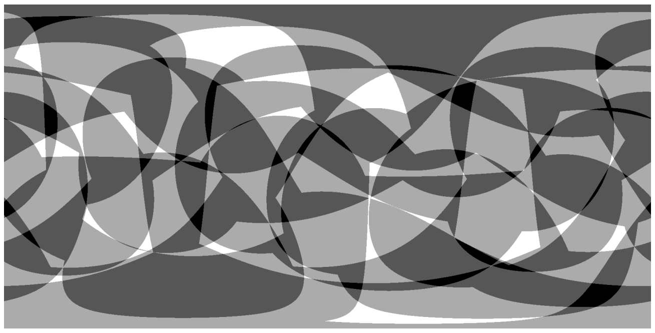


Fig. 6. Overlap for our 16 camera system (equirectangular). Black is 5x, white is 8x

potential overlap. As the focal length decreases, the image circle also decreases, reducing the number of pixels subtended by the optical FOV and hence reducing the PPD. On the other hand, it improves the overlap, since less of the image circle gets cropped above and below the sensor.

The camera count also influences the overlap. The trade-off between image circle size (i.e., PPD), camera count and overlap quality is shown in Table 1, which depicts some of the rig designs that were considered. Using this table, one can determine the number of cameras required to achieve a desired combination of angular resolution and overlap. Since we were seeking a good balance between camera overlap, spatial resolution, and camera count, we chose a middle ground of 16 cameras and 35 PPD, with close to 6x camera overlap. Figure 6 shows the overlap for each point in a rectilinear image. Note how most of the shell is seen by 6 or 7 cameras. More precisely, the fractions of the shell that are seen by 5, 6, 7, and 8 cameras are 3%, 50%, 43%, and 4%, respectively.

5 CAMERA CALIBRATION

The importance of calibration in depth estimation and reconstruction is often overlooked. Accurate calibration is essential for fast and accurate matching, and is also critical when combining reconstructed data with computer generated imagery (CGI). While great care was made to construct an extremely rugged, integrated, and precise camera rig, the actual observed images deviate from the original CAD model due to manufacturing variability. Even a small angular error of 0.1° results in a 3.5 pixel error at 35 PPD. Similarly, the lens principal point being off by 0.1 mm (100 microns) results in a 27 pixel deviation for a 3.65 micron pixel pitch.

Given that even small changes in the intrinsic and extrinsic camera parameters are significant, other mechanical variations during

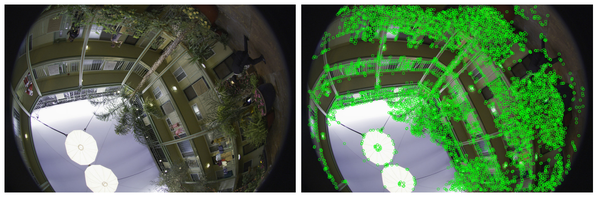


Fig. 7. Sample calibration image from the camera (left) and the same image with the extracted 2D features highlighted (right)

use become important, such as temperature changes, hardware maintenance or upgrades in the field, and rough on-set rig handling.

This demands an on-line, in-the-field calibration procedure that is automatic and removes the need for targets or specialized procedures. Fortunately, this requirement is less onerous than it sounds. The high degree of overlap allows for marker-less calibration using Structure from Motion (SfM) techniques similar to [Barazzetti et al. 2011; Slama 1980]. This is different from other existing systems that need a calibration room specifically design for this purpose, such as the Panono spherical camera [Khoramshahi and Honkavaara 2018]. Instead of fiducial markers, we find 3D features by matching 2D features across images, as described in Section 5.1. These 3D features, along with the ideal rig parameters, are used as input to our bundle adjustment implementation, simultaneously finding optimal calibrated values for the rig parameters and the 3D feature locations. We use a standard robust bundle adjustment algorithm [Triggs et al. 1999] implemented with the Ceres Solver [Agarwal et al. 2012]. Our final calibrated rig description updates camera position, orientation, focal length, and principal point, while leaving the nominal radial distortion parameters obtained from our lens simulation software as fixed.

5.1 Feature Extraction and Matching

The first step in 3D feature extraction is finding 2D features in the images. For each image, we create a full octave (half resolution) image pyramid. At each scale, we use OpenCV library functions to produce a list of features resolved to sub-pixel accuracy [Bradski 2000] and ignore any features that cannot be refined to sub-pixel accuracy. We then check that any features detected at multiple scales are not included more than once in the final list.

Next, we find feature correspondences between pairs of images. During the matching phase, we use the nominal (pre-calibration) rig parameters to direct our search for matches. We use the relative intrinsic and extrinsic geometries to pre-warp image patches in neighboring images so that direct patch comparison becomes possible. We also create a search window around each epipolar line by merging small square regions sampled at a number of candidate depth values along the epipolar line.

Features are compared using zero mean normalized cross correlation (ZNCC) to generate a score. A pair of features, a reference feature f_r and an overlap feature f_o, with ZNCC score s, are matched if the following criteria are satisfied:

(1) The ZNCC match score is above a threshold of $s_t = 0.75$.

(2) Neither feature has a higher ZNCC score with some other feature.
(3) Neither feature has a ZNCC score with some other feature that is within $\Delta = 0.05$ of this pair's score. This rule helps remove poor features (e.g. blank walls) or features that come from repeating patterns.

The final step in 3D feature construction is feature track computation, i.e., finding correspondences between a single 3D point with 2D features from multiple cameras. This is implemented by constructing a pairwise match graph, where each 2D feature is treated as a node in the graph and each pairwise match is an edge. Once the graph is assembled, each connected subgraph is interpreted as a single 3D feature. Each putative 3D feature is then checked to ensure that each 2D feature comes from a unique camera. The final assembled 3D tracks, along their corresponding 2D features, are then used to run bundle adjustment.

6 3D RECONSTRUCTION / DEPTH ESTIMATION

Our depth estimation pipeline uses a coarse-to-fine, hierarchical patch matching approach [Barnes et al. 2009; Roma et al. 2002; Wu et al. 2008], with a number of extensions, which we describe in this Section.Our depth estimation generates disparity maps for each individual camera. [6] It begins by applying a winner-take-all approach at the coarsest level of an image pyramid to get initial estimates from an exhaustive search over the entire depth range. At each subsequent level it exploits camera mappings to pre-compute color projections (Section 6.3), and it applies spatial propagation (Section 6.4), followed by random proposals (Section 6.5), geometric consistency (Section 6.6), and temporal color bilateral and median filters (Section 6.7). The disparity map estimates are then propagated to the next (finer) level. It can also use foreground masks to improve estimation (Section 6.8). The combination of all these steps in this particular order result in accurate depth estimates, as explained in Section 6.9.

While our reconstruction pipeline is based on traditional coarse-to-fine, hierarchical patch matching, one could also apply one of the more recent machine learning-based approaches such as [Chabra et al. 2019; Donne and Geiger 2019; Mildenhall et al. 2019; Srinivasan et al. 2019; Tonioni et al. 2019; Xu et al. 2019; Yang et al. 2019; Yao et al. 2019b; Zhang et al. 2019]. Since we are open sourcing our multi-view stereo datasets, we hope that others will investigate whether further improvements could be obtained using these newer approaches.

6.1 Dynamic Pyramid Scaling

The pyramid approach allows us to perform more expensive operations at coarser levels (e.g., winner-takes-all) and use the finer levels to refine disparity estimates [Barnes et al. 2009]. When constructing the pyramid, instead of using a constant scale factor we compute our level scales by fitting a quadratic between the finest and the coarsest level in log space. This results in faster level traversal (fewer levels), while maintaining depth estimation accuracy. This is because once we are at a sufficiently fine level, there is less

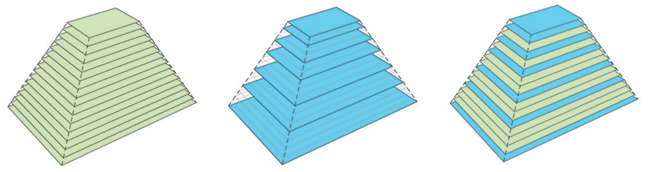


Fig. 8. Left to right: pyramid levels with a constant 0.9 scale, dynamic pyramid levels with scales in the range [0.9, 0.5], and both pyramids on top of each other for better comparison

information being added between levels compared to coarser levels. Figure 8 shows what the pyramid looks like when using a dynamic scaling approach. Our experiments show a 5x speedup on the entire depth estimation pipeline, with negligible disparity map differences when using dynamic pyramid scaling.

6.2 Cost Function

Our cost is a zero-mean sum of squared differences (ZSSD) between a reference color patch and all its overlapping camera patches. The average color is subtracted to account for lighting differences between cameras, which are non negligible when dealing with 360° inside-out rigs. Note that since our algorithm takes care of mapping cameras to each other and applying any necessary scaling and rotation, and because we use a small 3x3 patch window at all levels, ZSSD alone already does a good job comparing color patches and we do not need more complex metrics. We also compute a confidence weight to favor color patches with more detail. The confidence is just the inverse of the color variance of the reference patch $\sigma^2_{P_r}$, which is bounded below by an estimate of the image noise. Therefore, our total cost is:

$$c = \frac{1}{\sigma^2_{P_r}} \frac{1}{N-2} \sum_{i \in B_{N-2}} ZSSD(P_r, P_i), \tag{7}$$

where N is the number of overlapping cameras and B_{N-2} denotes the set of $N-2$ cameras after dropping the two worst scores.

6.3 Pre-computed Projections

When computing costs a square color patch in a reference image is compared against a set of overlapping camera color patches. When the cameras have different orientations, the patches need to be rotated so that they contain the same information, as shown in Figure 9. This is an expensive operation, e.g., with nine interpolations per overlapping camera in a 3x3 patch, and it happens at least once for every pixel on every camera at every level during depth estimation. This can be improved by pre-rectifying the overlapping camera images so that they align with the reference camera image. Note that the warp mapping between cameras only needs to be computed once for the entire sequence, since it is content independent. Figure 10 shows a set of aligned pre-computed projections. Even though the color projections need to be done potentially N^2 times at every level for every image in time, where N is the number of cameras, it is still less expensive than the original approach.

[6] We use the terms depth and disparity interchangeably, where disparity is the reciprocal of depth.

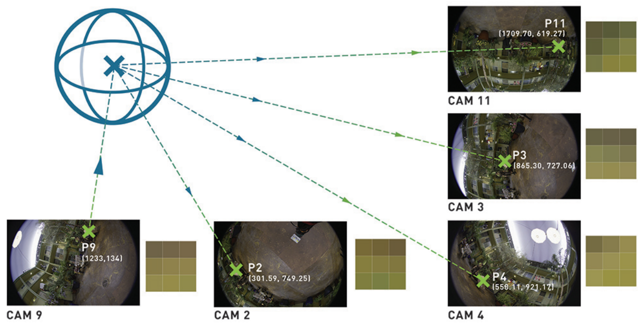


Fig. 9. Rotated color patches on overlapping cameras at the right disparity estimate

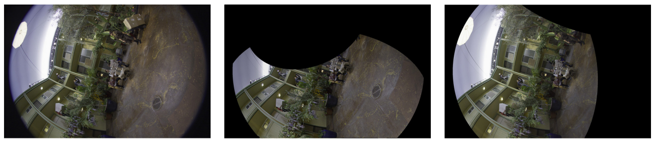


Fig. 10. Original color (left) and aligned projections of two overlapping cameras at infinity

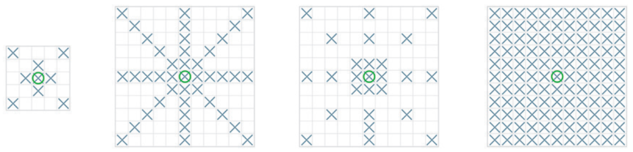


Fig. 11. Considered spatial propagation patterns. From left to right: near neighbors (chosen), asterisk, sparse asterisk, and dense block

6.4 Spatial Propagation

The spatial propagation in [Bleyer et al. 2011] describes how a good estimate can propagate its disparity across the image. However, this forces us to have a single threaded implementation since any pixel recursively depends on its neighbors' estimates. In order to parallelize the process the recursion can be replaced with a single set of look-ups according to a spatial pattern. Figure 11 shows some of the patterns considered. The larger the pattern the farther a good disparity estimate can be propagated, and the denser the pattern the more disparity candidates are considered. At the same time, the more samples the slower is the search. The runtime per pixel during cost computation increases linearly with the number of neighbors. Therefore, using a 5x5 near neighbor pattern instead of a dense block is 2.78 times faster. Our experiments showed that the "near neighbors" pattern gave us the best accuracy to runtime ratio, with negligible disparity map differences with respect to using a dense block pattern. Note that while these patterns are not capable of propagating estimates as far as the recursive methods similar results are achieved by applying this step at each level of the pyramid.

6.5 Random Proposals

Inspired by [Barnes et al. 2009; Schönberger et al. 2016; Zheng et al. 2014] we use random proposals to find disparity improvements, but we apply them to solve a different problem. High-frequency components of the color images are inevitably lost at coarser levels, and the initial winner-take-all approach is not able to estimate their depth. As we we move down the pyramid details start to appear and need to be re-evaluated. However, spatial propagation cannot resolve the new details because it is based on already existing disparities. A winner-take-all can be computationally expensive at finer levels, so we get around it by proposing several random disparities in a binary search fashion. For every random iteration if a proposed disparity d_p in the range $[d_{\min}, d_{\max}]$ results in a better cost than the current pixel's disparity, d_i, we reduce the search range to $[\min(d_i, d_p), \max(d_i, d_p)]$. Our experimental results show that thin structures are usually recovered using just two random proposals.

6.6 Geometric Consistency

When estimating depth using image-based techniques from overlapping camera images, it is common to encounter occlusions, especially around object edges [Bleyer et al. 2010; Kolmogorov and Zabih 2002; Woodford et al. 2009]. These occlusions are manifested by bad disparity estimates near depth discontinuities. These bad estimates, or mismatches, can be detected by looking for geometric inconsistencies between overlapping cameras, i.e. checking if other cameras see the same disparity at the same depth. Since the disparity of these mismatches mostly comes from neighbor proposals near edges, we can assume that the right disparity is somewhere farther away. Therefore, we replace a mismatched disparity with the median disparity of all the overlapping cameras that see something farther away. Figure 12 shows all the mismatches found in one of our disparity maps as well as the disparities we replaced those mismatches with. Most of the mismatches happen around the standing person and the ceiling lights, where there is a clear foreground separation.

6.7 Temporal Color Bilateral and Median Filters

We use a guided color bilateral filter [Caraffa et al. 2015; He et al. 2013] at each level to smooth the disparity maps while preserving the edges, as well as a median filter to get rid of disparity noise caused by bad estimates. In order to ameliorate temporal artifacts inherent in depth estimation, i.e. depth flickering, we extend our filter window over time, using both past and future frames [Bennett and McMillan 2005]. Figure 13 shows how our temporal filter can help resolve very fine details on moving objects.

6.8 Foreground Masks

It is often the case that a 360° capture is done without moving the camera rig, since it already covers the entire space around it. This can be leveraged by only estimating depth on objects that are moving, i.e. foreground pixels, and keeping background pixels untouched, which is related to ideas in *background maintenance and subtraction* [Criminisi et al. 2006; Toyama et al. 1999], as well as more recent

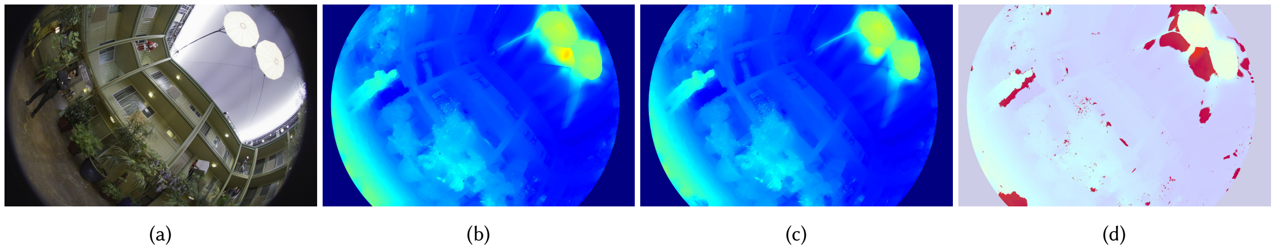


Fig. 12. Left to right: color image, disparity without geometric consistency, disparity with geometric consistency, and mismatched pixels before correction

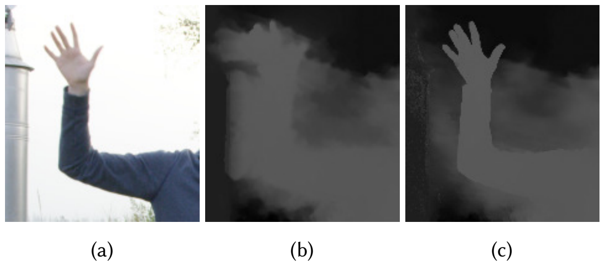


Fig. 13. Level 0 disparity without and with spatio-temporal filter. In the original sequence, the hand is waving; the fingers are resolved using samples over time

learning-based approaches to *video object segmentation* [Croitoru et al. 2019; Dutt Jain et al. 2017; Maninis et al. 2018; Yao et al. 2019a].

There are several benefits to this approach:

(1) Only a small portion of the pixels are potentially considered, which is directly proportional to the depth estimation runtime. In our tests we see a 50%-90% speedup with respect to not using foreground masks.
(2) Foreground pixels in a camera can be forced to only match foreground pixels in overlapping cameras. This greatly restricts the solution space and generally produces strictly better estimation. Note that this is true if foreground pixels are consistent between cameras, which is not usually the case when dealing with color noise.
(3) Foreground pixel occlusions are resolved, since pixels outside foreground objects are sent to the background. Note that multiple overlapping foreground objects do not have this benefit, unless the different foregrounds are captured separately.
(4) Depth is more stable over time, since static pixels are assigned a disparity when performing depth estimation on the background frame and kept for the entire sequence.

Note that we use foreground masks at every level to better propagate disparities. Figure 14 shows the final disparity map before and after applying the mask. Note how only 11.6% of the image is considered during depth estimation, and occlusions around foreground objects are resolved. Also, depth estimation is more than twice as fast. While our depth estimation pipeline includes a conventional background subtraction algorithm [Criminisi et al. 2006; Toyama et al. 1999], it allows the injection of any externally generated set of masks for flexibility.

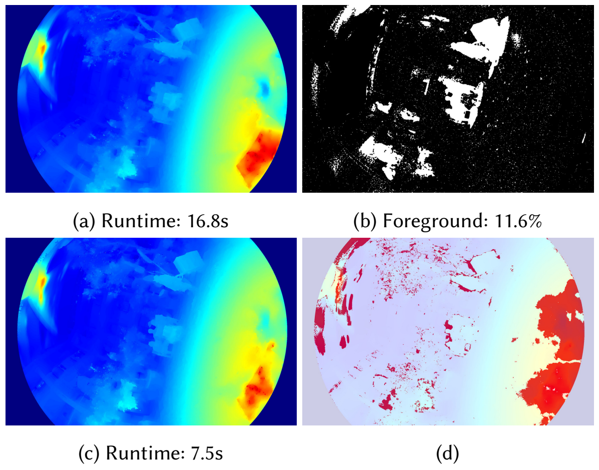


Fig. 14. Left to right, top to bottom: level 0 disparity without foreground masks, foreground binary mask, level 0 disparity using foreground masks, and disparity difference map

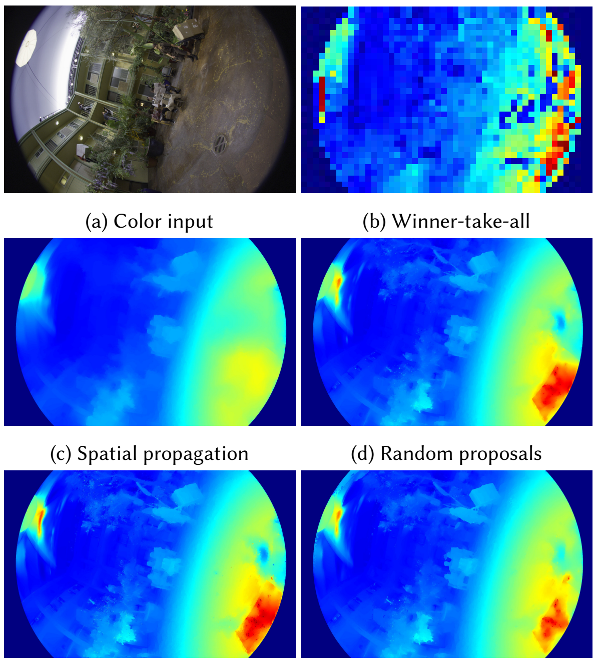
6.9 End-to-end Pipeline

As a summary, Figure 15 shows the additive contribution of all the steps performed during depth estimation. Note how the random proposals allow us to resolve small objects lost at coarser levels (e.g., tree leaves), as well as gaps between objects (e.g., space between the postman and his cart). Also note how we resolve most of the occlusions using background masks. The benefits of the temporal filter are better appreciated in the accompanying video.

7 RENDERING

The output of depth estimation is a separate depth map per color image. Given the depth map and the corresponding camera model, it is possible to project the color image outward to create a colorized point cloud corresponding to the 3D surfaces seen by the camera. This point cloud from a single camera will often resemble a lumpy, colorful canopy, as shown in Figure 16. To produce an image that represents the scene as seen from a novel viewpoint, we must combine all the point clouds.

The main challenge is reconciling contradictory and ambiguous depth data. One traditional approach is to resolve ambiguities offline by "fusing" the 16 point clouds into a single representation, and then to render the resulting geometry. Our renderer instead ingests



(a) Color input (b) Winner-take-all

(c) Spatial propagation (d) Random proposals

(e) Bilateral and median filter (f) Foreground masks

Fig. 15. Sequence of steps performed during depth estimation. (15b) Winner-take-all initializes disparity estimates to reasonable values at the coarsest level. (15c) Spatial propagation refines disparities based on existing estimates. (15d) Random proposals resolve small objects lost at coarser levels (e.g., tree leaves), as well as gaps between objects (e.g., space between the postman and his cart). (15e) Guided bilateral filtering smooths depths while maintaining sharp edges, and median filtering removes noise in the disparity map. (15f) Foreground masks resolve most of the remaining occlusions. A larger version of all these images can be found in the Supplemental Materials.

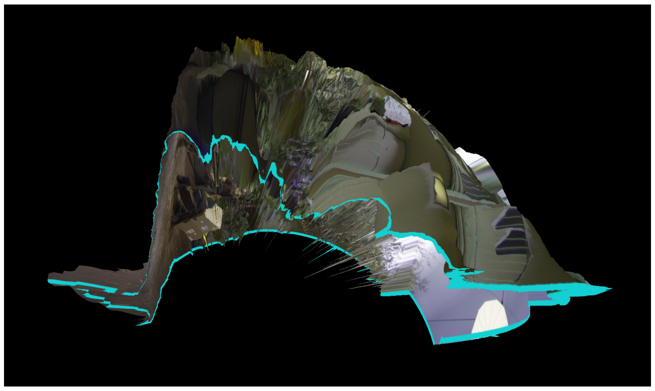


Fig. 16. A lumpy, colorful canopy resulting from projecting a camera's colored depth map into 3D

the separate per-camera depth and color images and performs the reconciliation in the GPU at render time.

7.1 Basic approach

We render each contributing point cloud as seen from the novel viewpoint separately. This is accomplished by rendering the depth map as a textured mesh, where the vertexes are derived by applying the appropriate calibrated camera model to the depth map. Because the point clouds are lumpy, overlap within a single point cloud, i.e., where the depth map folds on top of itself, is common and is resolved by using a standard depth test. In addition to rendering RGB by applying the color as a texture map, we also compute a weight (described in Section 7.2) that is stored in the alpha channel.

This produces 16 separate RGBA images, each representing the point cloud from a single camera's color and depth map as seen from the novel viewpoint. The final image is simply a weighted average of these images, and is performed on the GPU according to Algorithm 1. An accumulation buffer (*accum*) is used to keep a running total of weighted contributions (in *rgb*) and weights (in *a*) as each point cloud is processed. Note that the weighted blending

Algorithm 1 Combine all point clouds

```
$accum.rgba \leftarrow 0$
**for all** point clouds **do**
    render texture-mapped mesh w/ z-buffering $\longrightarrow cloud.rgba$
    $w \leftarrow k^{cloud.a}$
    $accum.a \leftarrow accum.a + w$
    $accum.rgb \leftarrow accum.rgb + w \cdot cloud.rgb$
**end for**
**return** $accumulate.rgb/accumulate.a$
```

stage does not use z-buffering. Instead, we assume that the occlusion relationships are correctly sorted out while rendering each warped input view. This is almost always the case, since our cameras have such wide fields of view that it is rare for an occluder visible in the new view to be out of view in one of the source images.

7.2 Weighting function

Multiple meshes will contribute to each pixel in the final image. We cannot simply average the meshes, as that would produce ghosting. Since the depth maps are mostly in agreement, a crisper image is obtained by letting certain contributions dominate the pixel.

To achieve this, we downweight by w_{cone} lower-quality pixels near the edge of each camera relative to higher-quality pixels near the center. Specifically, the profile of this weight is an elliptic cone:

$$w_{cone}(p) = \max(1 - |p - c|, \epsilon) \tag{8}$$

Another reason to down-weight a contribution is streaking. If the novel viewpoint happens to be exactly the field of view of one of the cameras, each pixel from that camera will correspond to exactly one pixel in the final image. Streaks will appear, however, if there is a discontinuity in the depth map and the novel viewpoint is moved to try to look behind that discontinuity, as shown in Figure 17. This is because the triangles that straddle the discontinuity are actually stretched between the near and the far depths. Only when seen from the original camera viewpoint do they provide reasonable information; when seen from the side, color and depth maps from other cameras should be preferred.

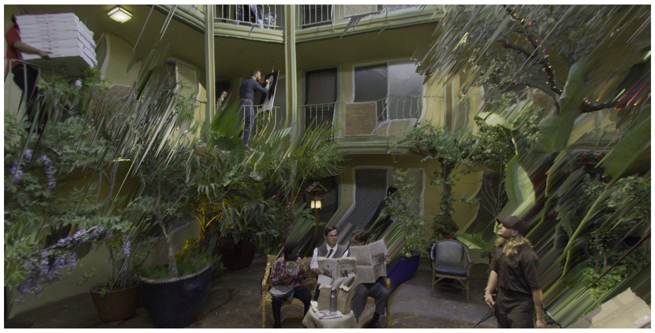


Fig. 17. Streaky triangles induced by depth discontinuities.

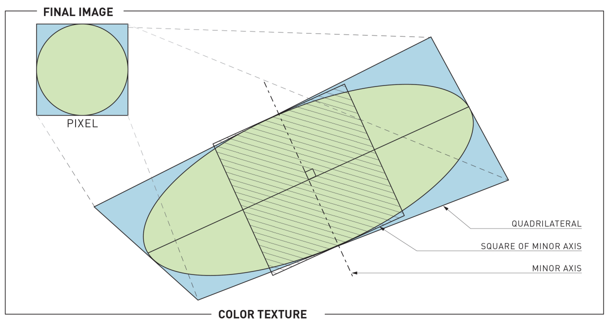


Fig. 18. Pixel and pre-image

To down-weight streaky triangles, we use the partial derivative of the texture coordinate that is used to look up the color [Kopf et al. 2014]. Recall that the pre-image of a circle inscribed in a pixel is an ellipse inscribed in a quadrilateral in the color texture, and that the sides of the quadrilateral are the partial derivatives, as shown in Figure 18. This ellipse represents the part of the color texture that contributes to the final pixel. Pixels from streaky triangles have smaller ellipses because they spread their allocated color texture across many pixels in the final image.

Weighting the pixels by the area of the quadrilateral (i.e., the Jacobian) might seem like a good idea. The minor axis also seems promising, as that represents the direction in which the streaky triangle is stretched. We are currently using the square of the minor axis (the shaded area in Figure 18) for w_{streak}, as this has proven to produce good results.

Finally, the combined down-weighting, w_i, is modified to achieve a sharper transition to reduce ghosting. We found that applying a softmax function with a large base produces a modified weight, w_i', with a transition that works well.

$$\begin{aligned} w_i &= w_{cone} w_{streak} \\ w_i' &= \frac{b^{w_i}}{\sum_j b^{w_j}}, \text{ where } b = e^{30} \end{aligned} \tag{9}$$

7.3 Background matting

Even when all point clouds have been rendered, some pixels are not properly accounted for. This happens, e.g., when the viewer leans far to one side to look behind a person in the scene. We fill in these pixels by using a matte which is simply the same background image that we used during depth estimation. This process is straightforward and surprisingly effective at removing artifacts that are otherwise jarring.

8 RESULTS

Our rendering pipeline generates a variety of different formats supporting multiple use case scenarios. Figure 19 shows the type of outputs we can generate, including cubemap and equirectangular projections, canopy snapshots, as well as combinations ready for VR headsets, such as left-right eye renders for omnistereo and RGBD maps for 3DoF. Figure 20 shows color and disparity snapshots of a 6DoF render at different head positions.

8.1 Depth Reconstruction

The depth reconstruction pipeline needs multi-camera overlap and sharp images in order to estimate disparities. The camera rig analyzer used to compute camera overlaps at multiple distances (e.g., Figure 6) can also be used to find the theoretical minimum distance at which at least one point in space is seen by just one camera. This number is 0.5 meters, which is closer than half the hyperfocal distance of 2 meters, where objects start to be acceptably sharp for reconstruction. A video provided in the Supplemental Materials shows color, disparity, and foreground mask maps of a person moving towards the camera. The quality of the disparity starts deteriorating as the subject gets closer than 1.6 meters, which matches our expectations.

8.2 Real-time playback

In order to achieve real-time 6DoF playback the color images and meshes may need to be resized. Using 180 degree lenses images are resized to 3k (3360x2160 in our case) to get an approximate effective 18 PPD allowed on modern VR headsets. Each camera image is then converted to 8-bit RGBA BC7 to achieve a 6:1 compression ratio, requiring a total of 5MB per image. The individual camera meshes are also simplified to 150k polygons, which translates to an extra 3MB per camera. This amounts to 3.8GB/s with a 16 camera rig at 30 frames per second. The player is hosted on a desktop computer with an NVIDIA GeForce GTX 1080 Ti GPU, and four RAID 0 NVMe drives to achieve a rate of 4 GB/s. A video showing the range of 6DoF motion and the appearance of objects at different distances can be found in the Supplemental Materials. Note that although technically the head translation is bounded by the diameter of the camera (1 meter in the video), the use of a background texture for static regions resolves occlusions and makes stepping out of the theoretical limit much more comfortable.

8.3 Rephotography Error

The perceptual image quality of the depth estimation results can be quantitatively measured by using a camera's original color as as reference, and reconstructing a rendering of that same camera using

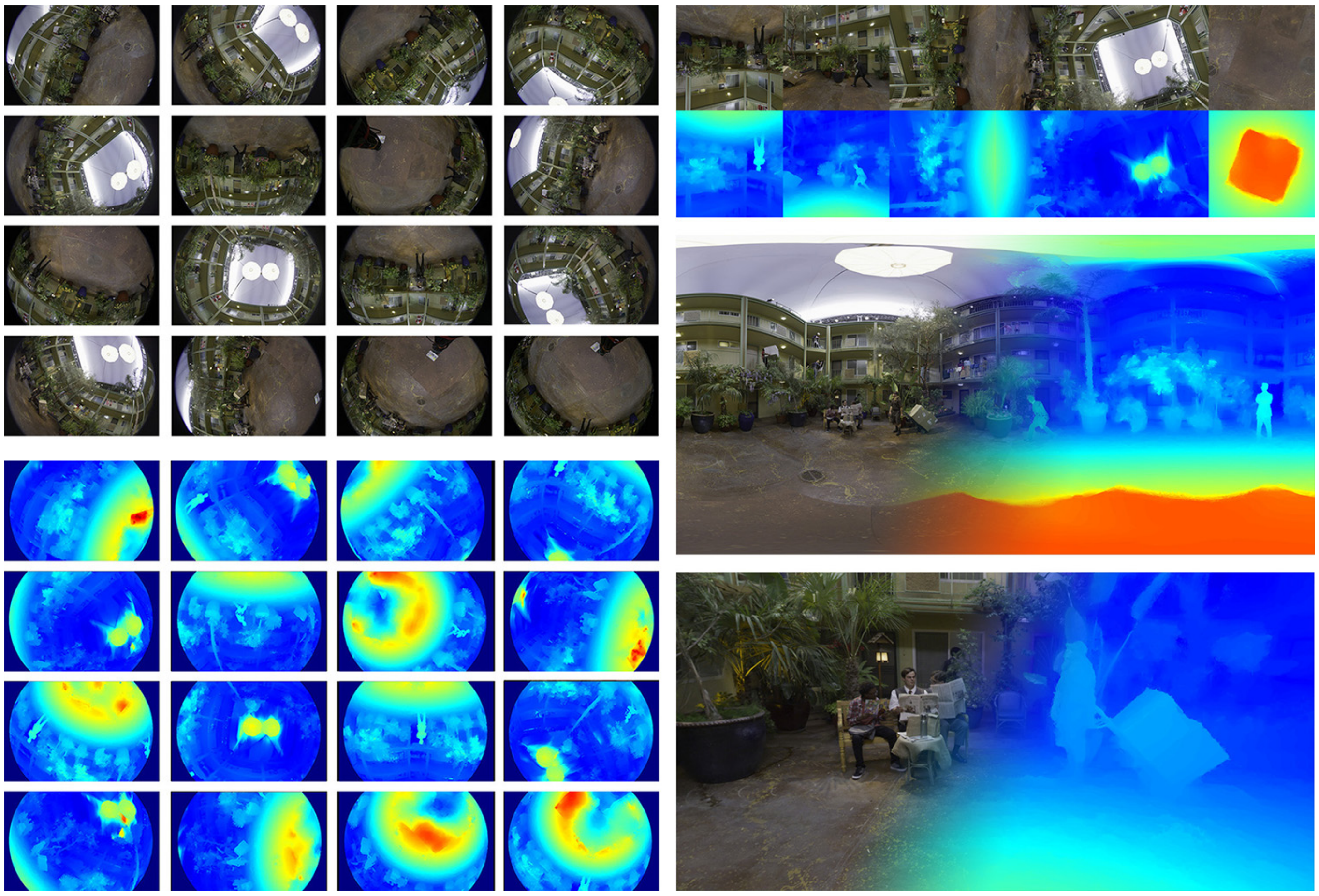


Fig. 19. Left: color images and estimated disparity maps. Right: top to bottom: cubemap color and disparity maps, equirectangular color and disparity maps, color and disparity canopy snapshots

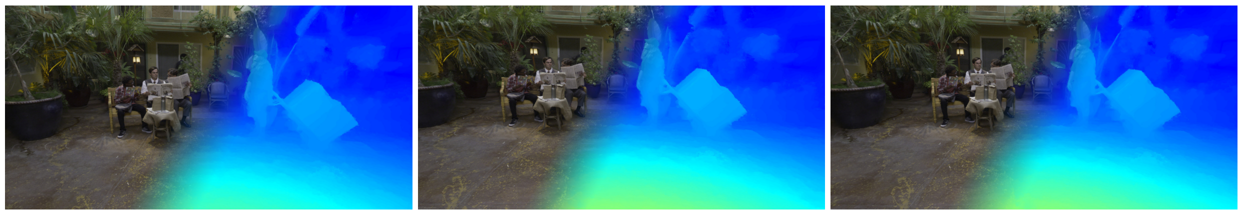

(a) Centered view (b) Negative x-axis translation (c) Positive x-axis translation

Fig. 20. Snapshots of 6DoF head movement. These are better observed in the accompanying video

the rest of the overlapping cameras, i.e., removing the reference camera. The per-channel mean structural similarity (MSSIM) between them gives us the rephotography error [Wang et al. 2004]. The resulting SSIM error maps can be visually examined to understand where the errors come from [Hedman et al. 2017; Waechter et al. 2017]. The MSSIM is computed in cubemap space to have uniform error coverage. The camera rig MSSIM is the mean MSSIM of all the cameras. Figure 21 shows the rephotography error results from one of the cameras. As can be seen, the errors are mostly focused around occlusions and small foliage. Our experiments show a median MSSIM of 91.75% on five datasets captured with a 16-camera rig with 180° FOV lenses, 8K sensors, and an average 30 mm baseline between cameras. These experiments were done without using foreground masks, since the reference camera color was removed, and it thus made sense to also discount all its associated data.

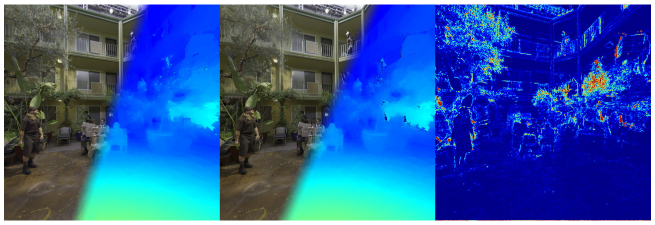


Fig. 21. From left to right: original color and estimated disparity, reconstructed color and disparity, rephotography error map. MSSIM = 93.37%. Blue: low error, red: high error

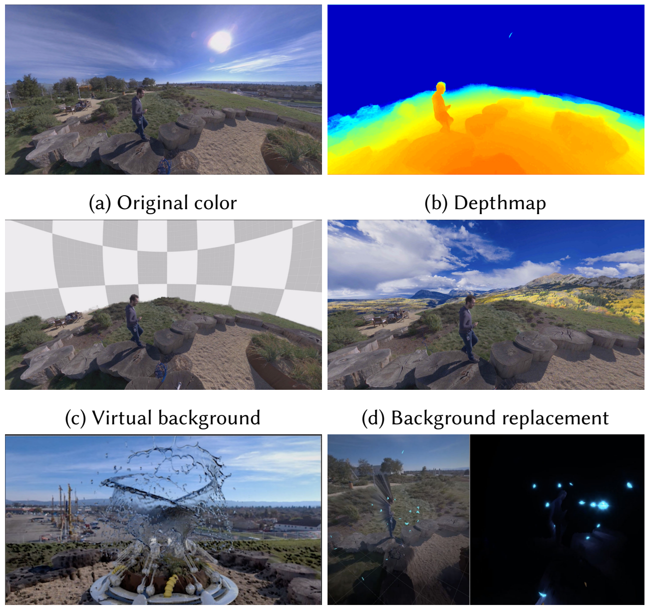


Fig. 22. Examples of compositing and relighting

8.4 Compositing and occlusion rendering

Given the accuracy and resolution of our depthmaps and the availability of color images from multiple angles, our system allows for high quality compositing, relighting, and occlusion rendering of the real-world footage with virtual content, as shown in Figure 22.

8.5 Runtime

Our tests show that a single frame from our 16 camera 8K rig is reconstructed in 1m 30s on a 6-Core Intel Xeon E5 at 3.5 GHz. However, our depth estimation pipeline is fully parallelizable at a frame level (including the temporal filtering), which means that we can use as many machines as needed to render. Our system supports local farm and cloud rendering, which allows us to render 2 minutes of 60 fps 8K footage overnight (10-12h) with 100 CPUs in AWS using 2.9GHz Intel Xeon instances. This is especially useful in a filmmaking environment, where it is imperative to be able to daily process content for the director and other crew members to view and assess the progress of the production.

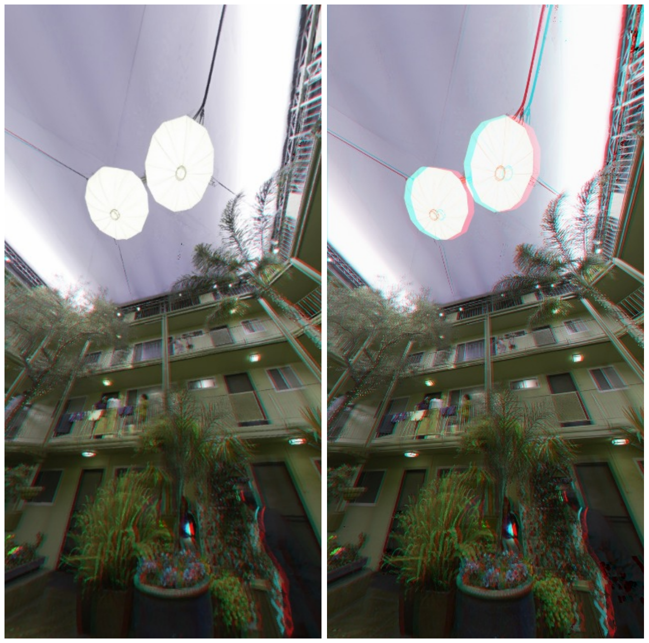


Fig. 23. Anaglyphs showing the difference between ODS and 6DoF as we look up. The effect can be seen using 3D glasses.

8.6 Comparison with ODS

As mentioned earlier, one limitation of ODS projections is they only support horizontal parallax and necessarily do not have stereo coverage near the poles. True 6DoF output overcomes this limitation with uniform parallax for all viewing directions and head tilt. Figure 23 illustrates this using a red/cyan anaglyph image that clearly shows how the stereo effect is lost towards the north pole in ODS but remains accurate in 6DoF.

9 CONCLUSION AND FUTURE WORK

We have designed, constructed, and commercially deployed a full 360° professional VR video capture system with an accompanying open source reconstruction and rendering pipeline. We published both the source code and a variety of data sets so that researchers and developers can build on this work and the output of the commercial cameras.

It is worth reiterating that even though we focused on the 16 camera RED system all our results are reproducible on other spherical and planar rigs. In fact we used various rigs during the development of our system. For instance, the images in Figure 22 was done with a 24 camera rig and the *ping pong*, *roadhouse* and *sheep* videos in the Supplemental Materials were shot with a planar 4 RED Helium camera proto-rig.

While this work was the culmination of many years of iterative development, in many ways it is just the beginning. There are numerous directions for further work.

As we designed the camera to take external genlock signals and timecode, we envision using a constellation of these cameras, say

5 cameras in a large room or set (e.g., one in the center and 4 near the edges), to fully capture an entire volume, and allow for truly unconstrained exploration of the reconstructed space in VR. Since the camera locations would be known, it would be possible to remove them during the reconstruction process. Interesting research challenges could include the fusion of the resulting multi-camera, depth augmented 360 videos, optimal placement of the cameras, and view interpolation algorithms that leverage recent advances in learning-based view synthesis.

Data movement dominates the majority of the post processing steps, so opportunities exist for exploiting coherence in the videos and associated depth for data compression and reduction of compute. Furthermore, the reconstruction algorithms, which currently represent the most expensive computational step, could readily be GPU accelerated and further improved.

We chose a geometric and depth map based approach to novel view synthesis because of the workflow and CGI advantages afforded by such an approach. However, this does not preclude using our camera and datasets for alternative image-based reconstruction, representations, and rendering algorithms.

The ability to reliably capture and reconstruct dynamic real-world environments opens doors for automating dataset generation for machine learning techniques, e.g., depth reconstruction, view synthesis, and potentially more sophisticated forms of dynamic scene understanding.

Prior to our work, the hurdle for capturing real-world environments for truly immersive VR experiences has been high. We believe that this paper significantly contributes to making such systems and required pipelines more widely accessible.

ACKNOWLEDGMENTS

The authors would like to thank the SIGGRAPH reviewers for the very helpful comments and useful requests for clarification. The Supplemental Video was produced by David Cherry and Knox Christopher - 100's hours were provided by their team on production shoots and post-production work. Jules Urbach and the whole OTOY team for support and cool effects images used in the paper. Johannes Kopf and Peter Hedman who provided insight and algorithmic improvements in and around depth estimation. The whole RED engineering and business team without which this camera could not have come into existence. Our content production team especially Eric Cheng who helped refine our workflow strategy. SuperSphere who performed exhaustive on-set production testing to help qualify the camera and give excellent on-set feedback. Colleen Henry who helped with early proto-types and was a tireless advocate and supporter of our effort. Finally, Brent Schnarr who spent countless hours in pre-, post- and on-set production work - none of the videos would have been possible without him.

REFERENCES

Hossein Afshari, Laurent Jacques, Luigi Bagnato, Alexandre Schmid, Pierre Vandergheynst, and Yusuf Leblebici. 2013. The PANOPTIC Camera: A Plenoptic Sensor with Real-Time Omnidirectional Capability. *Signal Processing Systems* 70, 3 (2013), 305–328.

Sameer Agarwal, Keir Mierle, et al. 2012. Ceres solver. (2012).

Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M. Seitz. 2016. Jump: Virtual Reality Video. *ACM Trans. Graph.* 35, 6, Article 198 (Nov. 2016), 13 pages.

Murat Aytekin and Michele Rucci. 2012. Motion parallax from microscopic head movements during visual fixation. *Vision research* 70 (2012), 7–17.

Luigi Barazzetti, Luigi Mussio, Fabio Remondino, and Marco Scaioni. 2011. Targetless camera calibration. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38, 5/W16 (2011), 8.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, Vol. 28. ACM, 24.

Eric P. Bennett and Leonard McMillan. 2005. Video Enhancement Using Per-pixel Virtual Exposures. *ACM Trans. Graph.* 24, 3 (July 2005), 845–852.

Tobias Bertel, Neill DF Campbell, and Christian Richardt. 2019. MegaParallax: Casual 360° Panoramas with Motion Parallax. *IEEE transactions on visualization and computer graphics* 25, 5 (2019), 1828–1835.

Michael Bleyer, Christoph Rhemann, and Carsten Rother. 2011. PatchMatch Stereo-Stereo Matching with Slanted Support Windows. In *Bmvc*, Vol. 11. 1–11.

Michael Bleyer, Carsten Rother, and Pushmeet Kohli. 2010. Surface stereo with soft segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1570–1577.

Gary Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. 2001. Unstructured Lumigraph Rendering. In *ACM SIGGRAPH 2001 Conference Proceedings*, Eugene Fiume (Ed.). ACM Press / ACM SIGGRAPH, 425–432.

Laurent Caraffa, Jean-Philippe Tarel, and Pierre Charbonnier. 2015. The Guided Bilateral Filter: When the Joint/Cross Bilateral Filter Becomes Robust. *IEEE Transactions on Image Processing* 24, 4 (April 2015), 1199–1208.

Rohan Chabra, Julian Straub, Christopher Sweeney, Richard Newcombe, and Henry Fuchs. 2019. StereoDRNet: Dilated Residual StereoNet. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

VR Circle. 2018. VR Movies in 360 Degree Virtual Reality. https://www.vrcircle.com/virtual-reality-360-degree-movies/. (2018). Accessed: 2019-05-18.

Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality Streamable Free-viewpoint Video. *ACM Trans. Graph.* 34, 4, Article 69 (July 2015), 13 pages.

Antonio Criminisi, Geoffrey Cross, Andrew Blake, and Vladimir Kolmogorov. 2006. Bilayer segmentation of live video. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 1. IEEE, 53–60.

Ioana Croitoru, Simion-Vlad Bogolin, and Marius Leordeanu. 2019. Unsupervised Learning of Foreground Object Segmentation. *International Journal of Computer Vision* 127, 9 (01 Sep 2019), 1279–1302.

Disney. 2008. Circle-Vision 360°. https://disney.fandom.com/wiki/Circle-Vision_360. (2008). Accessed: 2019-05-18.

Disney. 2016. Disney Movies VR. http://www.disneymoviesvr.com/. (2016). Accessed: 2019-05-18.

Simon Donne and Andreas Geiger. 2019. Learning Non-Volumetric Depth Fusion Using Successive Reprojections. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Suyog Dutt Jain, Bo Xiong, and Kristen Grauman. 2017. FusionSeg: Learning to Combine Motion and Appearance for Fully Automatic Segmentation of Generic Objects in Videos. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

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.

Kaiming He, Jian Sun, and Xiaoou Tang. 2013. Guided Image Filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 6 (June 2013), 1397–1409.

Eugene Hecht et al. 2002. *Optics*. Reading, Mass.: Addison-Wesley,.

Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. 2017. Casual 3D Photography. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 36, 6 (2017), 234:1–234:15.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 231.

Carlos Hernandez. 2016. Capture and share VR photos with Cardboard Camera, now on iOS. https://www.blog.google/products/cardboard/cardboard-camera-ios/. (2016).

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.

Ehsan Khoramshahi and Eija Honkavaara. 2018. Modelling and automated calibration of a general multi-projective camera. *The Photogrammetric Record* (Mar 2018), 86–112.

Vladimir Kolmogorov and Ramin Zabih. 2002. Multi-camera Scene Reconstruction via Graph Cuts. In *Proceedings of the 7th European Conference on Computer Vision-Part III (ECCV '02)*. Springer-Verlag, Berlin, Heidelberg, 82–96.

Robert Konrad, Donald G Dansereau, Aniq Masood, and Gordon Wetzstein. 2017. Spinvr: towards live-streaming 3d virtual reality video. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 209.

Johannes Kopf, Michael Cohen, and Richard Szeliski. 2014. First-person Hyperlapse Videos. *ACM Transactions on Graphics (Proc. SIGGRAPH 2014)* 33, 4 (August 2014).

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.

K-K Maninis, Sergi Caelles, Yuhua Chen, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. 2018. Video object segmentation without temporal information. *IEEE transactions on pattern analysis and machine intelligence* 41, 6 (2018), 1515–1530.

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.

Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM Trans. Graph.* 38, 4, Article 29 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3322980

Tim Milliron, Chrissy Szczupak, and Orin Green. 2017. Hallelujah: The World's First Lytro VR Experience. In *ACM SIGGRAPH 2017 VR Village (SIGGRAPH '17)*. ACM, Article 7, 2 pages.

Ryan S. Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. 2018. A System for Acquiring, Processing, and Rendering Panoramic Light Field Stills for Virtual Reality. *ACM Trans. Graph.* 37, 6, Article 197 (Dec. 2018), 15 pages.

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.

Eric Penner and Li Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 235.

Christian Richardt, Yael Pritch, Henning Zimmer, and Alexander Sorkine-Hornung. 2013. Megastereo: Constructing High-Resolution Stereo Panoramas. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2013)* (2013), 1256–1263.

Wesley JM Ridgway and Alexei F Cheviakov. 2018. An iterative procedure for finding locally and globally optimal arrangements of particles on the unit sphere. *Computer Physics Communications* 233 (2018), 84–109.

Nuno Roma, José Santos-Victor, and José Tomé. 2002. A Comparative Analysis Of Cross-Correlation Matching Algorithms Using a Pyramidal Resolution Approach. (05 2002).

Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 501–518.

Christopher Schroers, Jean Charles Bazin, and Alexander Sorkine-Hornung. 2018. An Omnistereoscopic Video Pipeline for Capture and Display of Real-World VR. *ACM Trans. Graph.* 37, 3 (2018), 37:1–37:13. https://dl.acm.org/citation.cfm?id=3225150

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.

Chester C Slama. 1980. *Manual of Photogrammetry.* Technical Report. America Society of Photogrammetry,.

Pratul P. Srinivasan, Richard Tucker, Jonathan T. Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the Boundaries of View Extrapolation With Multiplane Images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jayant Thatte, Jean-Baptiste Boin, Haricharan Lakshman, and Bernd Girod. 2016. Depth augmented stereo panorama for cinematic virtual reality with head-motion parallax. In *IEEE International Conference on Multimedia and Expo, ICME 2016, Seattle, WA, USA, July 11-15, 2016*. 1–6.

Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. 2019. Real-Time Self-Adaptive Deep Stereo. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kentaro Toyama, John Krumm, Barry Brumitt, and Brian Meyers. 1999. Wallflower: Principles and Practice of Background Maintenance. In *Seventh International Conference on Computer Vision (ICCV'99)*. 255–261.

Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. 1999. Bundle Adjustment — A Modern Synthesis. In *International Workshop on Vision Algorithms*. Springer, 298–372.

Matthew Uyttendaele, Antonio Criminisi, Sing Bing Kang, Simon Winder, Richard Hartley, and Richard Szeliski. 2004. Image-Based Interactive Exploration of Real-World Environments. *IEEE Computer Graphics and Applications* 24, 3 (May/June 2004), 52–63.

Michael Waechter, Mate Beljan, Simon Fuhrmann, Nils Moehrle, Johannes Kopf, and Michael Goesele. 2017. Virtual Rephotography: Novel View Prediction Error for 3D Reconstruction. *ACM Trans. Graph.* 36, 1, Article 45a (Jan. 2017).

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612.

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.

Eric W. Weisstein. 1998. Thomson Problem. (1998). http://mathworld.wolfram.com/ThomsonProblem.html Visited on 19/05/16.

Oliver Woodford, Philip Torr, Ian Reid, and Andrew Fitzgibbon. 2009. Global Stereo Reconstruction under Second-Order Smoothness Priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31, 12 (Dec 2009), 2115–2128.

Changchang Wu, B. Clipp, Xiaowei Li, J. Frahm, and M. Pollefeys. 2008. 3D model matching with Viewpoint-Invariant Patches (VIP). In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 1–8.

Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. 2019. Deep View Synthesis from Sparse Photometric Images. *ACM Trans. Graph.* 38, 4, Article 76 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3323007

Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. 2019. Hierarchical Deep Stereo Matching on High-Resolution Images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Rui Yao, Guosheng Lin, Shixiong Xia, Jiaqi Zhao, and Yong Zhou. 2019a. Video Object Segmentation and Tracking: A Survey. *arXiv preprint arXiv:1904.09172* (2019).

Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. 2019b. Recurrent MVSNet for High-Resolution Multi-View Stereo Depth Inference. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip H.S. Torr. 2019. GA-Net: Guided Aggregation Net for End-To-End Stereo Matching. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. 2014. PatchMatch Based Joint View Selection and Depthmap Estimation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)*. IEEE Computer Society, Washington, DC, USA, 1510–1517.

C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. 2004. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics (Proc. SIGGRAPH 2004)* 23, 3 (August 2004), 600–608.