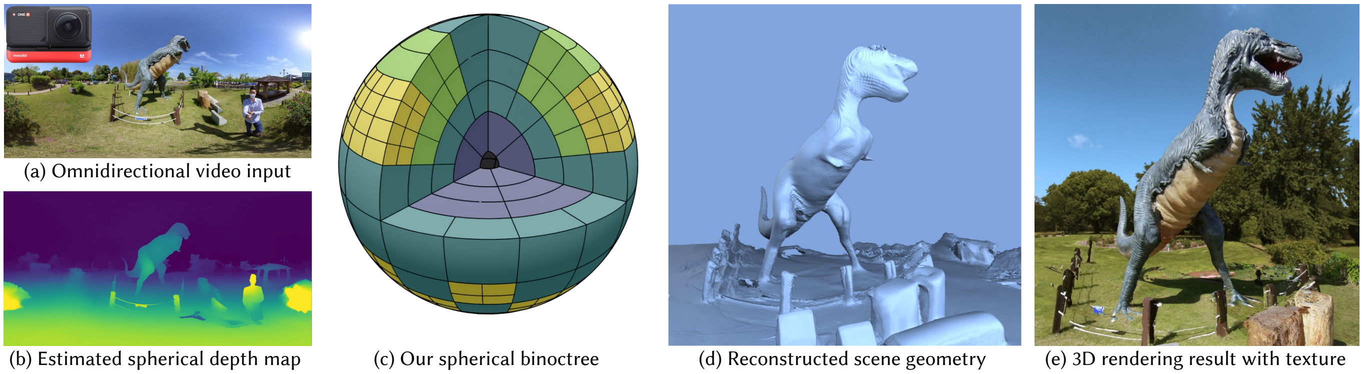
Egocentric Scene Reconstruction from an Omnidirectional Video

HYEONJOONG JANG, KAIST, South Korea
ANDRÉAS MEULEMAN, KAIST, South Korea
DAHYUN KANG, KAIST, South Korea
DONGGUN KIM, KAIST, South Korea
CHRISTIAN RICHARDT, University of Bath, United Kingdom
MIN H. KIM, KAIST, South Korea


(a) Omnidirectional video input

(b) Estimated spherical depth map

(c) Our spherical binoctree

(d) Reconstructed scene geometry

(e) 3D rendering result with texture

Fig. 1. We introduce a practical reconstruction method for 3D scene geometry from short handheld omnidirectional videos. (a) Example video frame captured by a 360° camera (inset). (b) An inverse depth frame estimated by our spherical disparity estimation. (c) To reconstruct egocentric scene geometry effectively from a short omnidirectional video, we devise a scene reconstruction method using a novel *spherical binoctree* data structure. (d) The reconstructed 3D scene geometry. (e) 3D rendering of the reconstructed scene with our texture mapping. Please see our supplemental video for additional results and comparisons.

Omnidirectional videos capture environmental scenes effectively, but they have rarely been used for geometry reconstruction. In this work, we propose an egocentric 3D reconstruction method that can acquire scene geometry with high accuracy from a short egocentric omnidirectional video. To this end, we first estimate per-frame depth using a spherical disparity network. We then fuse per-frame depth estimates into a novel *spherical binoctree* data structure that is specifically designed to tolerate spherical depth estimation errors. By subdividing the spherical space into binary tree and octree nodes that represent spherical frustums adaptively, the spherical binoctree effectively enables egocentric surface geometry reconstruction for environmental scenes while simultaneously assigning high-resolution nodes for closely observed surfaces. This allows to reconstruct an entire scene from a short video captured with a small camera trajectory. Experimental results validate the effectiveness and accuracy of our approach for reconstructing the 3D geometry of environmental scenes from short egocentric omnidirectional video inputs. We further demonstrate various applications using a conventional omnidirectional camera, including novel-view synthesis, object insertion, and relighting of scenes using reconstructed 3D models with texture.

CCS Concepts: • **Computing methodologies → 3D imaging**.

Additional Key Words and Phrases: 360° video, 3D reconstruction, binoctree, spherical disparity, TSDF fusion

ACM Reference Format:
Hyeonjoong Jang, Andréas Meuleman, Dahyun Kang, Donggun Kim, Christian Richardt, and Min H. Kim. 2022. Egocentric Scene Reconstruction from an Omnidirectional Video. *ACM Trans. Graph.* 41, 4, Article 100 (July 2022), 12 pages. https://doi.org/10.1145/3528223.3530074

1 INTRODUCTION

Omnidirectional cameras with two fisheye lenses (e.g., Ricoh Theta or Insta360) have become popular in recent years as they are cost-effective for capturing spherical images and videos of surrounding scenes. This makes these imaging devices ideal for many vision applications, such as SLAM. Spherical images have been commonly used in this regard of omnidirectional imaging, but rarely for geometric understanding or reconstruction of 3D scenes.

In traditional 3D reconstruction, depth maps are obtained by multi-view stereo algorithms that are heavily specialized for perspective images. Depth maps are then often fused using truncated signed distance functions stored in a Cartesian voxel grid. This data structure is effective particularly for *object-* or *room-scale*, perspective inputs. However, as we employ spherical image inputs that capture *outward-looking* (egocentric) spherical views where distant surfaces have less details, observations should be handled at different resolutions depending on the distance between the camera center and surfaces. In other words, the data structure for egocentric

Authors' addresses: Hyeonjoong Jang, KAIST, South Korea, hjjang@vclab.kaist.ac.kr; Andréas Meuleman, KAIST, South Korea, ameuleman@vclab.kaist.ac.kr; Dahyun Kang, KAIST, South Korea, dhkang@vclab.kaist.ac.kr; Donggun Kim, KAIST, South Korea, dgkim@vclab.kaist.ac.kr; Christian Richardt, University of Bath, United Kingdom, christian@richardt.name; Min H. Kim, KAIST, South Korea, minhkim@kaist.ac.kr.

0730-0301/2022/7-ART100
https://doi.org/10.1145/3528223.3530074

spherical input should be parameterized by the distance from the camera, resulting in a spherical coordinate system. We therefore devise a spherical subdivision scheme for 3D reconstruction that is specially designed for omnidirectional cameras.

In this work, we present a practical 3D reconstruction method that acquires *scene-scale* geometry with texture from an egocentric, handheld omnidirectional video captured with a short camera trajectory. To this end, we introduce a new approach for dense spherical depth estimation from spherical stereo images using a spherical disparity network trained on a new synthetic spherical RGBD video dataset. We then devise a novel hierarchical data structure, the *spherical binoctree*, which subdivides the spherical space into binary and octree nodes that represent spherical frustums adaptively to assign high-resolution nodes for closely observed surfaces, as shown in Figure 1c. We also combine our spherical binoctree with an adaptive truncated signed distance function designed to tolerate errors in the spherical depth estimated from the given handheld video. To reconstruct a high-resolution global texture map, we devise a texture reconstruction method that selects high-resolution texture information from near frames by evaluating input frames with respect to the reconstructed surfaces in a regularized resolution of solid angles. Lastly, our complete 3D reconstruction system enables various applications using scene-scale geometry, such as novel view synthesis, relighting, and placing virtual objects into the reconstructed scene. We make our source code available for reproducibility and also release our new synthetic spherical RGBD video dataset on our project page https://vclab.kaist.ac.kr/siggraph2022p2/.

2 RELATED WORK

Spherical Depth Estimation. Methods for monocular spherical depth estimation [Eder et al. 2019; Jiang et al. 2021; Jin et al. 2020; Pintore et al. 2021; Sun et al. 2021; Wang et al. 2020b; Zeng et al. 2020; Zioulis et al. 2019, 2018] and learned spherical stereo methods [Lai et al. 2019; Wang et al. 2018, 2020a] are mostly trained on synthetic indoor scenes, and thus tend to perform poorly on real and/or outdoor scenes. Spherical rectification [Li 2008; Matzen et al. 2017] enables the use of traditional correspondence finding methods, which we exploit in our approach. Spherical multi-view stereo often builds on sphere sweeping [Im et al. 2016; Komatsu et al. 2020; Meuleman et al. 2021; Won et al. 2019a,b] and is often limited in depth map resolution. None of these methods is specialized for 3D reconstruction from spherical input. On the other hand, we combine an optical flow network with spherical rectification to estimate spherical depth. Please see Supplemental Section 1 for a more detailed discussion of spherical depth estimation techniques.

Mesh-based Reconstruction. Sparse scene geometry, for example from structure-from-motion (SfM) or simultaneous localization and mapping (SLAM), can be reconstructed as a mesh by triangulation of a point cloud [Kang and Szeliski 1997] or by fitting a proxy geometry [Bertel et al. 2020]. More geometric detail can be recovered by meshing dense depth maps computed using stereo or multi-view stereo [Pollefeys et al. 2004]. Kim and Hilton [2013] reconstruct multiple meshes from spherical stereo images at discrete locations, and merge them into a global mesh using surface selection. Several view synthesis methods estimate per-view depth maps using multi-view stereo and blend them together at render time [Hedman et al. 2016; Overbeck et al. 2018; Parra Pozo et al. 2019], which does not produce a consistent global scene geometry. Other approaches stitch the per-view depth maps into a panoramic depth map that is converted into a multi-layer mesh for handling occlusions [Hedman et al. 2017; Hedman and Kopf 2018; Serrano et al. 2019; Zhang et al. 2020]. While this works well for view synthesis, no complete geometry model is reconstructed. In contrast, our method enables efficient reconstruction of complete scene geometry.

Volumetric Reconstruction. Fusing multiple depth maps into globally consistent scene geometry is greatly simplified using a volumetric representation, such as the truncated signed distance function (TSDF). Curless and Levoy [1996] introduced this approach for combining depth maps from a laser range finder, and KinectFusion [Izadi et al. 2011] popularized this approach for active depth sensors. However, the scale and resolution of the scene geometry is limited by the use of a dense voxel grid. Follow-up work therefore exploited sparsity using voxel hashing [Nießner et al. 2013] or an octree-based voxel grid [Zeng et al. 2013], to reconstruct high-quality meshes of larger scenes. These methods are mostly limited to indoor scenes due to the short depth range and small field-of-view of the depth sensors, which makes scanning large scenes difficult, particularly outdoors. Won et al. [2020] passively reconstruct large indoor scenes by integrating multiple spherical depth maps into a TSDF volume using voxel hashing. This requires a lot of memory as near and far objects are stored using the same fixed voxel size. Lidar imaging can provide reliable but sparse depth outdoors [Kühner and Kümmerle 2020], but is still expensive. None of these techniques can perform scene-scale 3D reconstruction from spherical images. To the best of our knowledge, our approach is the first specially designed for short trajectories of an omnidirectional video camera.

3 SPHERICAL SCENE RECONSTRUCTION

Acquisition Setup. We capture an omnidirectional video using a conventional 360° video camera, an Insta360 ONE R, which can capture videos with 5760×2880 resolution at 30 fps. Because it is an omnidirectional video, it is unavoidable to capture the photographer. However, we found that when using a monopod, the photographer can easily be masked out using a static mask image. Our videos are 10–120 seconds in length and we show results for different camera motions, including circular motions and along more general curves.

Camera Pose Estimation. We estimate per-frame camera poses using OpenVSLAM [Sumikura et al. 2019], an omnidirectional visual SLAM method that has since been discontinued by the authors. A suitable alternative would be OpenMVG's omnidirectional structure-from-motion [Moulon et al. 2016]. Following the approach of Bertel et al. [2020], we run the SLAM twice for better pose accuracy. In the first pass, we extract feature points, estimate rough camera poses and reconstruct a sparse 3D point cloud. Due to the progressive optimization, the estimated camera poses do not reach the required accuracy. In the second pass, we reconstruct the camera poses again based on the previously reconstructed sparse 3D map.

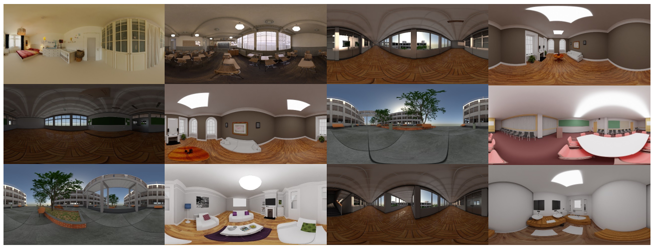


Fig. 2. Frames from the training set of our spherical RGBD video dataset.

3.1 Spherical Depth Estimation

For a given omnidirectional video, we first estimate per-frame depth maps using a learning-based spherical stereo technique. We adapt an existing perspective optical flow estimation network to learn spherical disparity estimation, and fine-tune it on our new spherical RGBD video dataset. We then combine multiple stereo pairs for each video frame to remove outliers and mask unreliable areas, for robust spherical depth estimation.

3.1.1 Spherical RGBD Video Dataset. To achieve more accurate depth maps than existing spherical depth estimation networks [Komatsu et al. 2020; Wang et al. 2020a; Won et al. 2019a], we fine-tune an off-the-shelf optical flow estimation network trained for perspective images using a new synthetic dataset of spherical videos with ground-truth depth maps and camera poses (see Figure 2). We collect twelve diverse scenes [Blender Online Community 2022; McGuire 2017], from which we render a total of 102 upright spherical RGBD videos using Blender. Each video is 500 frames long and rendered in equirectangular format at a resolution of 2048×1024 pixels. The camera is moving along a random 3D spline path.

3.1.2 Spherical Rectification. Like for perspective images, spherical images can be rectified, so that finding correspondence becomes a 1D search problem for the disparity along epipolar lines. We use the spherical rectification technique by Li [2008], which conveniently aligns epipolar lines with horizontal image scanlines. First, we rotate both spherical images to align their cameras' x-axes with the baseline between the cameras (see Figure 3a). Then, we rotate the second image about the x-axis such that corresponding axes of both camera coordinate systems are parallel to each other. The resulting spherical images are then represented using *transverse equirectangular* projection [Matzen et al. 2017], in which the (epi)poles are along the left and right image edges, as shown in Figure 4. Once rectified, we can apply any correspondence algorithm, including optical flow, as described in the next section.

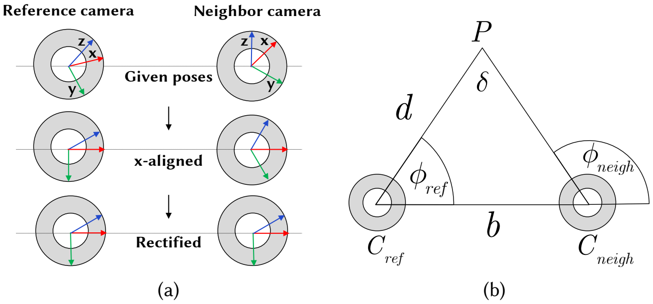
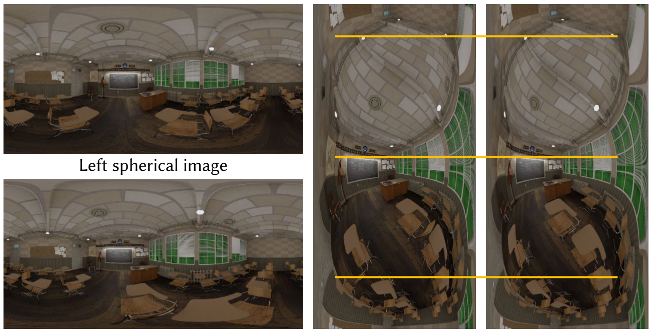


Fig. 3. (a) Steps of spherical rectification according to Li [2008]. (b) For a stereo pair of cameras C_{ref} and C_{neigh} with baseline b, a 3D point P is projected onto each spherical image at angles ϕ_{ref} and ϕ_{neigh}, respectively, which defines the spherical disparity δ of the point P.



Left spherical image

Right spherical image

Rectified spherical stereo pair

Fig. 4. Epipolar lines (orange) become horizontal after spherical rectification.

3.1.3 Spherical Disparity vs. Distance. The geometric relation between distance and disparity for omnidirectional cameras is different from that of a perspective camera. To supervise our spherical disparity estimation network, we need to convert per-pixel ground-truth (radial) distance to spherical disparity in the transverse equirectangular projection of a rectified spherical stereo pair (Figure 3a).

We derive the disparity using the notation in Figure 3b and use it for both reconstruction and rendering. According to the law of sines, we can relate the angular disparity δ at the point P to the baseline b between cameras, the angle $(\pi - \delta - \phi_{\text{ref}})$ at the neighbor camera C_{neigh}, and the distance d of the point P from the reference camera C_{ref}:

$$\frac{\sin\delta}{b} = \frac{\sin(\pi - \delta - \phi_{\text{ref}})}{d} = \frac{\sin(\delta + \phi_{\text{ref}})}{d} \tag{1}$$

$$= \frac{\sin\delta\cos\phi_{\text{ref}} + \cos\delta\sin\phi_{\text{ref}}}{d}. \tag{2}$$

Next, we rearrange to solve for the disparity δ:

$$\frac{d\sin\delta}{b} = \sin\delta\cos\phi_{\text{ref}} + \cos\delta\sin\phi_{\text{ref}}, \tag{3}$$

$$\sin\delta\cdot\left(\frac{d}{b} - \cos\phi_{\text{ref}}\right) = \cos\delta\sin\phi_{\text{ref}}, \tag{4}$$

$$\frac{\sin\delta}{\cos\delta} = \tan\delta = \frac{\sin\phi_{\text{ref}}}{\frac{d}{b} - \cos\phi_{\text{ref}}} = \frac{b\sin\phi_{\text{ref}}}{d - b\cos\phi_{\text{ref}}}, \tag{5}$$

$$\delta = \arctan\left(\frac{b\sin\phi_{\text{ref}}}{d - b\cos\phi_{\text{ref}}}\right). \tag{6}$$

The angular disparity δ is then linearly scaled to disparity $\Delta = \frac{w\delta}{\pi}$ in the transverse equirectangular projection based on the image width w (in pixels).

3.1.4 Disparity Estimation. We build our spherical disparity estimation on RAFT [Teed and Deng 2020], which has recently set a new state of the art in optical flow estimation between *perspective* images. We adapt RAFT for horizontal disparity estimation between rectified spherical stereo images by restricting the vertical flow component to zero, i.e., we only optimize for the horizontal component of the flow that corresponds to disparity. We then fine-tune the parameters of the original perspective model on our new custom-made spherical RGBD video dataset. As illustrated in Figure 5, this adaptation step helps to overcome the discrepancy between the perspective images originally used for training RAFT and the transverse equirectangular images with spherical distortions.

During training, we randomly pick a stereo pair from our spherical RGBD video dataset, so that various camera baselines are used for training. Then, for the selected stereo pair, we rectify both stereo images and compute the ground-truth disparity map from the ground-truth depth using Equation 6. Due to memory limitations of our NVIDIA Titan RTX GPU, we had to reduce the input spatial resolution from 1024×2048 to 768×1536. For the training, we optimized 1,000 iterations with batch size 8, using 8,000 stereo pairs in total.

Using the fine-tuned model, we estimate $K = 11$ disparity maps from the selected neighbor frames in Section 3.1.5. Each stereo pair is rectified independently and padded by one-eighth of its height at the top and bottom edges using wrap-around padding. In this way, we can achieve better continuity of the spherical image during correspondence finding. Next, we estimate stereo disparity from the reference to the neighbor view and crop the resulting disparity maps back to the unpadded size. Then, we convert each disparity map to a depth map using the inverse of Equation 6. Finally, we apply the validity masks described above to ignore unreliable regions, and merge all K depth maps into one depth map that is both more accurate and more robust, using the per-pixel median of the valid pixels to remove outliers.

3.1.5 Neighbor Frame Selection. Our goal is to estimate per-frame depth maps for every input frame. For each frame, we thus select K neighboring frames that are used to form K stereo pairs for disparity estimation. This frame selection is critical for estimating disparity with high accuracy, as this depends on the baseline between cameras. If the baseline is too short, depth estimation becomes ill-conditioned, particularly for far objects. On the other hand, wide baselines can also result in unreliable depth estimates, because of increasing occlusions and deteriorating correspondence quality. We trade off these considerations by selecting the $K = 11$ temporally closest neighboring frames whose baselines are larger than a minimum baseline threshold. This parameter depends on the scene and camera path. We determine it by searching for the minimum baseline at which a third of the absolute estimated disparities exceed 5 pixels.

3.1.6 Validity Mask Generation. We apply two binary masks to exclude unreliable depth estimates. The first mask covers the photographer, which would produce incorrect depth estimates and is not the target of scene reconstruction. We select these areas manually, but they could also be segmented automatically. The second mask covers the left and right edges of the transverse equirectangular image as depth estimation is ill-conditioned near the epipoles, which are located at the left and right image edges due to the spherical rectification. We mask out 15% of pixels by width on both the left and the right edges. We do not include occlusion masks for two reasons: (1) our disparity estimation network is already trained in an occlusion-aware manner; and (2) masking out all occluded pixels can lead to insufficient depth estimates in regions that are only observed in the reference frame.

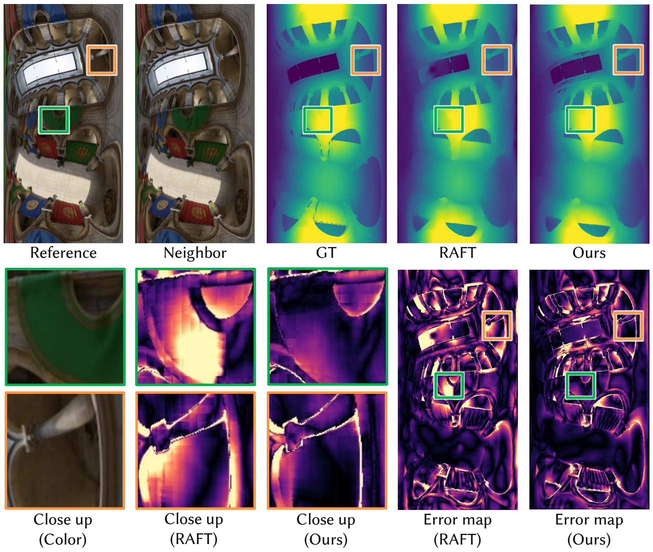


Fig. 5. Accuracy of our spherical disparity estimation on a test scene. Top: Input spherical stereo pair, ground-truth disparity map (GT), and estimates by RAFT [Teed and Deng 2020] and our adapted model. Bottom: Close-up crops of the reference image and absolute disparity error maps (the darker, the better) on the right. Original RAFT shows less accurate disparity (e.g., see green crop) and struggles with highly distorted regions (e.g., see orange crop), which our adapted model can handle better.

3.2 Spherical Scene Geometry Reconstruction

Given the per-frame spherical depth maps reconstructed in the previous section, the next step is to extract the surface geometry of the scene. The point clouds obtained from per-view depth maps can be used to incrementally reconstruct a surface [Litvinov and Lhuillier 2013], or fused and denoised [Wolff et al. 2016]. However, the most popular approach is to fuse all depth maps into a truncated signed distance function (TSDF) stored in a Cartesian voxel grid [Curless and Levoy 1996], and to extract the level set surface using marching cubes [Lorensen and Cline 1987]. However, a voxel grid becomes inefficient when the camera trajectory is relatively small compared to the captured scene, because everything is divided into same-sized voxels, even far-away surfaces that do not require such a fine resolution. Octrees reduce memory usage by assigning voxels only where sparse surfaces exist [Zeng et al. 2013]. However, the size of voxels at the deepest level of the octree is still the same.

To solve this problem, we introduce the *spherical binoctree* – a data structure specifically tailored for reconstructing scenes from multiple spherical depth maps. Each node of the spherical binoctree stores a TSDF value at its center, and we dynamically allocate nodes by subdivision when a 3D point is newly observed. We combine both binary and octree subdivision in one data structure – hence the name 'binoctree'. Finally, we generate a triangle mesh using dual marching cubes [Schaefer and Warren 2005], which extends marching cubes to octrees.

3.2.1 Spherical Binoctree Structure. We define our spherical binoctree using spherical coordinates $(\phi, \theta, 1/r)$ instead of Cartesian coordinates (x, y, z), where r is the radial distance from O_{tree}, the center of the octree. Using inverse depth, we can represent the full range of depth, from nearby to infinitely far away, like the sky. For simplicity, we explain our spherical binoctree structure in terms of depth. The octree is centered at O_{tree}, which is the average of all camera centers. The extent of space covered by the octree is defined by three ranges: $0 \le \phi < 2\pi$, $0 \le \theta \le \pi$, $r_{\text{near}} \le r < r_{\text{far}}$, where ϕ represents azimuth, θ the polar angle, and r the radius (distance from O_{tree}), respectively. Each node of the octree represents a specific *spherical frustum* that is defined by six values: $(\phi_{\min}, \theta_{\min}, r_{\min})$ and $(\phi_{\max}, \theta_{\max}, r_{\max})$. At the root level, we subdivide space into octants by splitting the azimuth angle ϕ into four intervals, $\left[0, \frac{\pi}{2}\right)$, $\left[\frac{\pi}{2}, \pi\right)$, $\left[\pi, \frac{3\pi}{2}\right)$, $\left[\frac{3\pi}{2}, 2\pi\right)$, the polar angle into two intervals, $\left[0, \frac{\pi}{2}\right)$ and $\left[\frac{\pi}{2}, \pi\right]$, and keeping the radius undivided.

3.2.2 Sphere Volume Division. Our spherical binoctree data structure is a mixture of binary and octree subdivision of a sphere's volume. Our initial spherical binoctree only contains eight nodes, which require further (potentially recursive) subdivision *on demand* according to the reconstructed spherical depth maps. For each valid pixel in each depth map, we consider the corresponding 3D point P and, if necessary, subdivide the octree node it falls into recursively, to ensure appropriate spatial resolution from all input camera viewpoints. As our spherical binoctree mixes angular (ϕ, θ) and spatial dimensions (r), repeated subdivision can lead to *elongated nodes* in the radial direction. We address these cases using an alternative binary subdivision. See Figure 6 for an illustration of the spherical binoctree and our subdivision policies.

Binary Radial Subdivision Policy. In normal octrees, all axes represent spatial dimensions, but our ϕ and θ axes are angular dimensions. In this case, repeated subdivision produces increasingly elongated nodes as the angular extent of nodes shrinks faster than their radial length. To prevent spherical frustums from becoming too elongated, and to ensure their shapes stay *balanced*, we introduce a binary subdivision step along the radial dimension (see Figure 6c). We check if a spherical frustum is balanced using the condition:

$$1.4 \times \underbrace{(\phi_{\max}-\phi_{\min})\left(\frac{r_{\min}+r_{\max}}{2}\right)}_{\text{arc length at } r=(r_{\min}+r_{\max})/2} < \underbrace{r_{\max}-r_{\min}}_{\text{radial extent}}, \qquad (7)$$

where the constant 1.4 encourages more isotropic node shapes via subdivision. Note that the symbols in Equations 7 and 8 refer to the extent of a given spherical frustum. If a node is imbalanced, we split it into two along the radial direction only. This step is repeated until the node corresponding to the point P is balanced.

Eightfold Subdivision Policy. Assuming the node containing the point P is balanced, we next check whether it is sufficiently small when observed from the camera's viewpoint at C_{ref}. For this, we approximate the spherical frustum, which has a volume of

$$\begin{aligned} V_{\text{node}} &= \int_{\phi_{\min}}^{\phi_{\max}} \int_{\theta_{\min}}^{\theta_{\max}} \int_{r_{\min}}^{r_{\max}} r^2 \sin\theta \; dr\, d\theta\, d\phi \qquad (8) \\ &= \frac{1}{3}\left(r_{\max}^3 - r_{\min}^3\right)(\cos\theta_{\min} - \cos\theta_{\max})(\phi_{\max}-\phi_{\min}), \end{aligned}$$

as a sphere of equal volume, which has a radius of $t = \sqrt[3]{3V_{\text{node}}/4\pi}$ and subtends a solid angle of

$$\Omega = 4\pi \sin^2\left(\frac{\alpha}{2}\right) \quad \text{for} \quad \alpha = \sin^{-1}\left(\frac{t}{d_{\text{node}}}\right), \qquad (9)$$

where d_{node} is the distance between the center of the node and the capturing camera. While the solid angle Ω is larger than a threshold T_{solid}, we subdivide the node into eight subnodes. For most scenes, we use $T_{\text{solid}} = 0.0001$ sr. We split the angular extents of the node in the middle, and select the geometric mean of $r_{\min}$ and $r_{\max}$ in the radial direction, such that the ratio of the lengths of the inner and outer arcs remains the same.

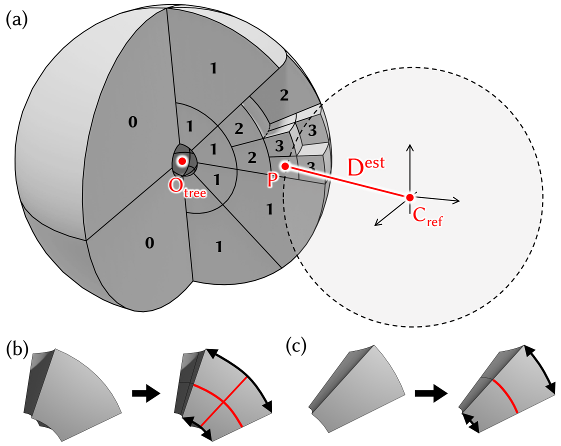
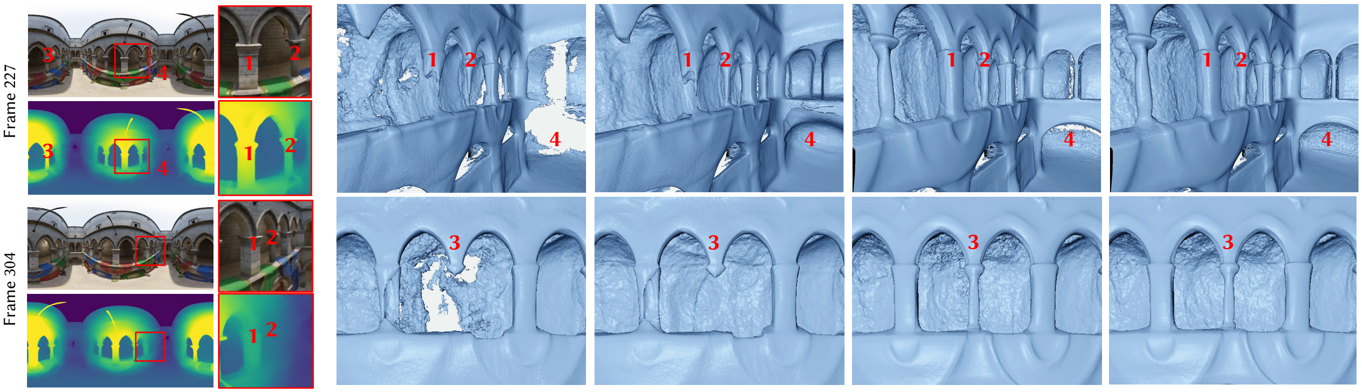


Fig. 6. Our spherical binoctree and subdivision schemes: (a) Illustration of our spherical binoctree with numbers in each node representing the tree depth from the root of the octree (located at O_{tree}). The size of a node is a function of the distance D^{est} between an observed point P and the center C_{ref} of the capturing camera. (b) Balanced nodes are divided into eight nodes. (c) Imbalanced (elongated) nodes are only divided radially.

3.2.3 TSDF Integration. After generating an appropriately subdivided spherical binoctree for every single depth estimate in the spherical depth maps, we gather all the leaf nodes and start updating TSDF values. We make two key modifications to the standard TSDF update procedure [Curless and Levoy 1996] to improve the quality of the final geometry reconstruction. First, we adapt the truncation threshold depending on depth, which takes into account that depth estimation errors increase proportional to depth. Second, we introduce a confidence-based weighted TSDF update step that considers depth and color consistency. We implemented the TSDF integration in parallel on the GPU, as all operations can be performed independently.

Depth-dependent Truncation Threshold. A fixed truncation threshold cannot handle the increasing depth error in far regions without sacrificing quality in close regions. This can lead to large holes in the geometry, as shown by the wall labeled "4" in Figure 7b. To solve this problem, we adaptively increase the truncation threshold T_{trunc} as a function of estimated depth D^{est} (see Figure 7c):

$$T_{\text{trunc}}(D^{\text{est}}) = e_{\text{m}} D^{\text{est}} + e_{\text{n}}, \qquad (10)$$



(a) RGB input and the reconstructed depth map

(b) Constant truncation w/o confidence weight

(c) Adaptive truncation w/o confidence weight

(d) Constant truncation w/ confidence weight

(e) Adaptive truncation w/ confidence weight

Fig. 7. The effect of confidence-based weighting using w_{update} and adaptive truncation thresholds when updating TSDF values in our spherical binoctree. The input video contains 1000 frames along a vertical elliptical trajectory in the Sponza scene. (a) Color images and estimated depth maps for frames 227 and 304. (b)–(e) Meshes reconstructed by our algorithm using all input frames, with different weighting and truncation thresholds. See Supp. Table 3 for error metrics.

where e_{m} and e_{n} are slope and offset parameters that are empirically determined to account for the camera trajectory with respect to the scene. Note that this truncation threshold linearly increases as the depth error increases due to the spatial resolution of the camera.

To update the TSDF values stored in the spherical binoctree, we project the centers of all leaf nodes into each depth map and average the SDF values if a valid depth value exists at the pixel p on the depth map, and if the estimated depth D^{est} exceeds the distance to the center of the node minus the truncation threshold T_{trunc}. In other words, we do not update TSDF values if the node center is more distant than estimated by more than the threshold T_{trunc}.

Confidence-based Weighting. Real scenes often contain dynamic objects, reflections and texture-less regions, which can be problematic when integrating depth estimates from different viewpoints or timestamps. To handle these problems, we introduce Gaussian-like weights for proximity, depth consistency, and color consistency. We penalize viewpoints that are far away and thus capture a point at lower resolution, based on the estimated depth $D_i^{\mathrm{est}}(p)$ for a pixel p in camera i as follows:

$$w_{\mathrm{p}}(i,p) = \exp\left(-\left(D_i^{\mathrm{est}}(p)\right)^2/\sigma_{\mathrm{p}}\right), \tag{11}$$

where the parameter σ_{p} depends on scene scale and camera trajectory. Larger σ_{p} result in an increased contribution of depth values from far cameras to the overall weight.

Our depth consistency weight considers the estimated depth $D_i^{\mathrm{est}}(p)$ for pixel p in camera i, and its depth in a neighboring view $j \in N(i)$ after reprojection using $\pi_{i\to j}(p)$. However, in most cases, the distance of a point to both cameras will be different. What is the same, though, is the distance of the point to the line through the cameras, which we calculate using

$$\widetilde{D}_i^{\mathrm{est}}(p) = D_i^{\mathrm{est}}(p)\sin\phi_{\mathrm{rect}}(p), \tag{12}$$

where $\phi_{\mathrm{rect}}(p)$ is the azimuth angle of p in the rectified stereo pair.

To penalize when the reference distance $\widetilde{D}_i^{\mathrm{est}}(p)$ and the projected distance $\widetilde{D}_j^{\mathrm{est}}(\pi_{i\to j}(p))$ from the neighbor view j are too different, we define the depth consistency weight as

$$w_{\mathrm{d}}(i,j,p) = \exp\left(-\left(1-\frac{\widetilde{D}_i^{\mathrm{est}}(p)}{\widetilde{D}_j^{\mathrm{est}}(\pi_{i\to j}(p))}\right)^2/\sigma_{\mathrm{d}}\right). \tag{13}$$

This weight also handles dynamic objects within a scene. The value of σ_{d} varies per scene as the distance reprojection error is affected by the reconstructed scale of the scene. We set σ_{d} to the mean of the first frame's depth map, divided by 1000.

Finally, we quantify color consistency using

$$w_{\mathrm{c}}(i,j,p) = \exp\left(-\left\|I_i(p)-I_j(\pi_{i\to j}(p))\right\|_2^4/\sigma_{\mathrm{c}}\right), \tag{14}$$

which penalizes large color mismatches more strongly. We use $\sigma_{\mathrm{c}} = 24$ for 8-bit color values across all scenes.

The final weight for our TSDF update combines weights from all neighboring frames of frame i:

$$w_{\mathrm{update}}(i,p) = w_{\mathrm{p}}(i,p)\cdot\sum_{j\in N(i)} w_{\mathrm{d}}(i,j,p)\cdot w_{\mathrm{c}}(i,j,p). \tag{15}$$

Figure 7 evaluates the impact of this weight on 3D reconstruction.

3.2.4 Surface Mesh Extraction.

Finally, we extract a triangular mesh from the TSDF stored in our spherical binoctree. The irregular octree subdivision and curved shape of nodes prevent the direct application of marching cubes [Lorensen and Cline 1987]. However, Schaefer and Warren [2005] introduced marching cubes on the dual graph of an octree by connecting the zero-crossing points between adjacent nodes, regardless of their size or shape. Their approach also generalizes to our spherical binoctree as its dual graph is well-defined, and TSDF values are stored at the center of each node.

3.3 Texture Atlas Reconstruction

Once we obtain the 3D mesh geometry from our spherical binoctree, we generate a single texture map based on a tiled projective texture reconstruction approach. In our acquisition setup, the reconstructed surfaces are usually visible in more than one frame in the input video. Therefore, our method searches input frames for the optimal view for each texture tile using a winner-take-all approach that we adapt to the specific constraints of our spherical scene reconstruction. We reconstruct a tile-based texture map for the surface from neighboring spherical input views by considering occlusions and visibility when blending textures.

Since our spherical binoctree is designed to make each node subtend a similar solid angle relative to the input cameras (Section 3.2.2), all triangles in the reconstructed mesh have approximately the same size in terms of solid angle. As the solid angle is proportional to the amount of color information captured at a particular viewpoint, we allocate a fixed extent of the texture map to each triangle face while preserving the acquired texture details across the full range of object distances in the scene. Figure 8b shows an example of the equilateral triangle tiling texture map.

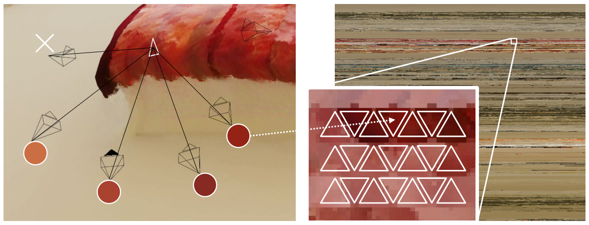
3.3.1 Equilateral Mesh Unwrapping. Different from popular texture atlas reconstruction scenarios with perspective input images [Zhou and Koltun 2014], our spherical texture reconstruction should handle a wide range of distortion of input images to the global texture space associated with the spherical geometry. Therefore, we adapt a tile-based texture data structure [Lee et al. 2020] and specially customize it to the spherical domain.

To this end, we introduce a mesh unwrapping method using equilateral triangle tiling with sufficiently low complexity to make it practical even for large outdoor scenes, while correctly handling the irregular sizes of binoctree nodes. In our geometry reconstruction pipeline, a surface far from the camera, such as a distant building, will be subdivided less and thus result in a larger mesh face. However, it is not necessary to represent its texture at a high resolution, since the captured color information of distant surfaces is relatively low-resolution. Instead, we use a tiling of uniformly sized equilateral triangles as a texture map, which avoids allocating excessive space for large but distant faces [Lévy et al. 2002] or complicated geometric parametrization [Sander et al. 2002; Zhou et al. 2004].

Each pixel in each triangle's texture is potentially seen in every single input video frame, resulting in a huge number of potential combinations that are impractical to check completely. Therefore, we make two simplifying assumptions. First, we select a subset of 5–20 representative input frames instead of using all N input frames. We do this using k-means clustering of the camera poses and selecting the center frames to ensure sufficient observations to handle occlusion well. Second, we assume that each triangle's texture originates from exactly one input view, and we treat the center of the triangle as the representative for the entire triangle's texture to reduce per-pixel computation.

3.3.2 Optimal Texture Selection. For each triangle, we want to select the texture from the optimal frame. To this end, we first check which representative views can see the triangle (i.e., without occlusion), and then calculate a visibility score for each view, to find the best one to sample the texture from.

Occlusion Check. We start by projecting the center of a particular triangle into each reference view, say to pixel p. We then check for occlusions by comparing the depth of the triangle center, $D^{\mathrm{tri}}(p)$, to the rendered depth map of the reconstructed mesh, $D^{\mathrm{rend}}(p)$, using the visibility ratio


(a) Visibility test with multiple camera viewpoints

(b) Our final texture atlas

Fig. 8. (a) The optimal frame is selected from multiple viewpoint candidates using a visibility score in our texture mapping. (b) Optimal textures fill the tiled equilateral triangle texture map.

$$V(p) = \frac{D^{\mathrm{tri}}(p)}{D^{\mathrm{rend}}(p)}. \tag{16}$$

This ratio is close to one if the triangle is visible, and it is greater than one if the triangle is occluded by a closer triangle, as this leads to a lower rendered depth $D^{\mathrm{rend}}(p)$ compared to the triangle's depth $D^{\mathrm{tri}}(p)$. We are scoring views differently depending on whether the triangle is visible or not, so that we can extract textures even for triangles that are not seen from any view.

Visibility Score. We choose the optimal view for texturing based on a combination of criteria: (1) the view's depth estimate should be consistent with the reconstructed mesh, also nearby, (2) closer viewpoints are preferable as they resolve textures more finely, and (3) observations closer to the normal direction minimize projective distortions. For efficiency, we compute all visibility scores for a view in parallel using a cube map. Considering the geometric error of our mesh, the visibility ratio V is only a weak indicator. Therefore, we define a binary mask M that encodes the consistency between rendered depth $D^{\mathrm{rend}}(p)$ and the depth $D^{\mathrm{est}}(p)$ estimated via our occlusion-aware spherical disparity network (Section 3.1):

$$M(p) = \mathbb{1}\left(0.9 < \frac{D^{\mathrm{est}}(p)}{D^{\mathrm{rend}}(p)} < 1.1\right), \tag{17}$$

where $\mathbb{1}$ is the indicator function. Regions with inconsistent depth, where $M(p) = 0$, should also reduce the score of nearby triangles. We implement this using a soft erosion operation:

$$M'(p) = \min(M(p), M^*(p)), \tag{18}$$

where M^* is a blurred version of the consistency mask M. We use a Gaussian blur with $\sigma = 13$ pixels for 960×960 cube-map faces. Our complete visibility score also relies on the triangle's normal $\mathbf{n}$, the direction $\mathbf{v}$ to the view, and the distance to the triangle $D^{\mathrm{tri}}(p)$:

$$S(p) = \frac{\mathbf{n} \cdot \mathbf{v}}{D^{\mathrm{tri}}(p)} \times \begin{cases} M'(p) & \text{if } V(p) \leq 1.02 \\ (M'(p) - 2)V(p) & \text{if } V(p) > 1.02 \end{cases} \tag{19}$$

Note that we compute different scores for visible (top) and hidden triangles (bottoms), which are non-negative and negative, respectively. This ensures that the best view is chosen first, even if the triangle is occluded in all views.

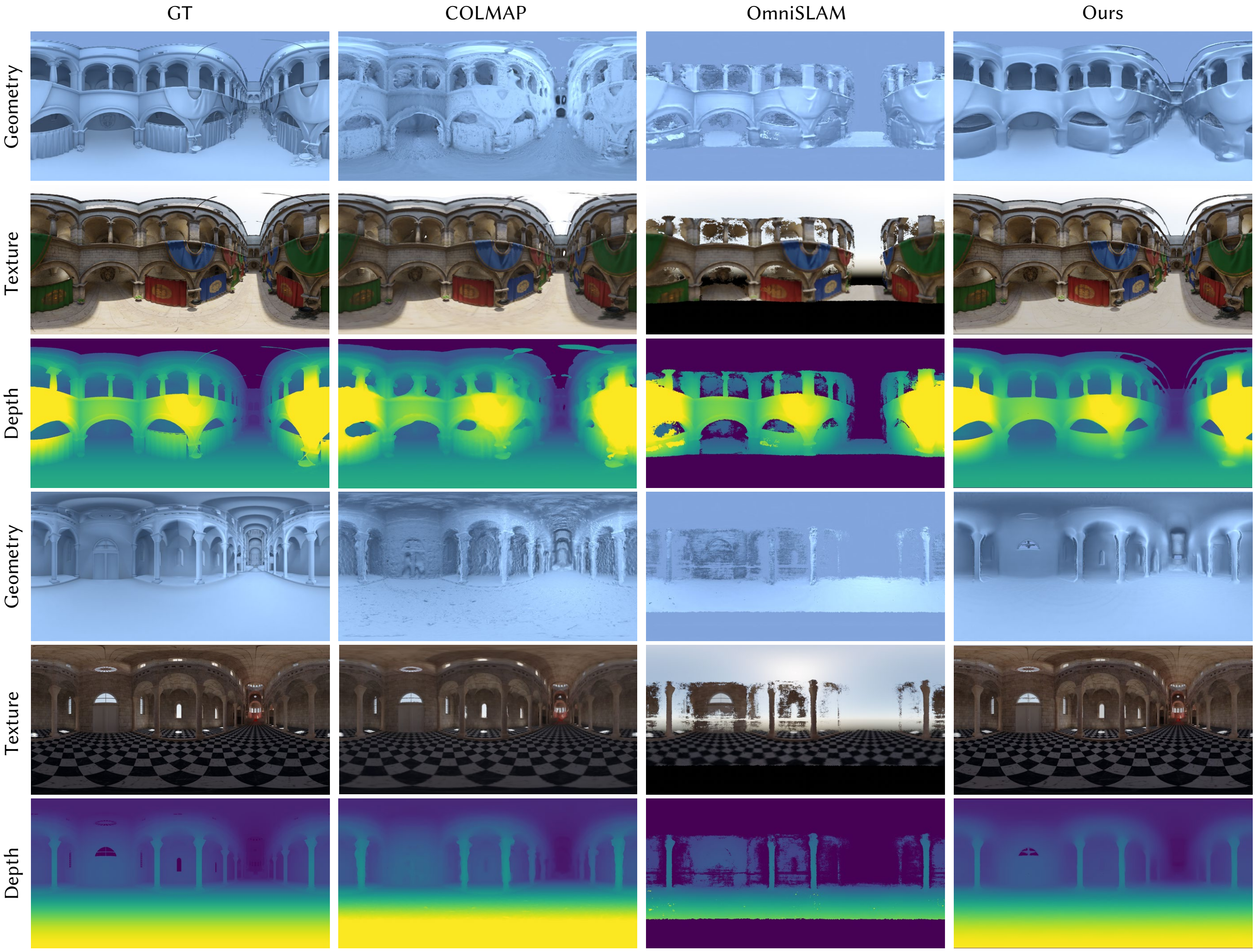


Fig. 9. Comparison of synthetic reconstruction accuracy with COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016] and OmniSLAM [Won et al. 2020] for central spherical views. Note that COLMAP's reconstruction shows artifacts (top), and OmniSLAM's reconstruction truncates the north/south poles and distant regions. Our reconstruction is the closest to the ground truth. Refer to Table 3 for depth and color error metrics.

4 RESULTS AND EVALUATION

4.1 Spherical Stereo Depth Accuracy

First, we evaluate the depth accuracy of our spherical stereo method. We compare two-view depth estimation accuracy with current [Teed and Deng 2020; Wang et al. 2020a] and classic methods [Hernandez-Juarez et al. 2016; Hirschmüller 2008]. For this, we rendered 100 rectified spherical stereo pairs with ground-truth depth at 512×1024 resolution from five 3D models not in our training dataset ('Sponza', 'Lone-monk', 'San Miguel', 'Pabellon_sunset', 'Sibenik Cathedral'). For each of the five models, we render five different stereo pairs at baselines of $\{10, 20, 30, 40\}$ cm. Figure 5 shows an example comparison with RAFT [Teed and Deng 2020]. In Table 1, we compare the mean absolute error (MAE), RMSE, and the percentage of bad pixels with an error of more than 0.1 and 0.4 in inverse depth (lower is better for all metrics). Our method outperforms all other methods in every measure. We refer to the supplemental document for a breakdown by baseline (Table 2) and more results (Section 2.1).

Table 1. Comparison of two-view spherical depth estimation methods and our method. The columns '>0.1' and '>0.4' show the percentage of bad pixels that exceed an absolute inverse depth error of 0.1/0.4 [m^{-1}]. Our method significantly outperforms all others.

Method	>0.1	>0.4	MAE	RMSE
360SD-Net [Wang et al. 2020a]	52.61	24.22	0.351	0.540
SGBM [Hirschmüller 2008]	15.21	4.66	0.070	0.186
GPU-SGM [Hernandez-Juarez et al. 2016]	16.17	5.12	0.082	0.210
RAFT [Teed and Deng 2020]	11.46	2.43	0.059	0.187
Ours	**7.97**	**0.55**	**0.035**	**0.075**

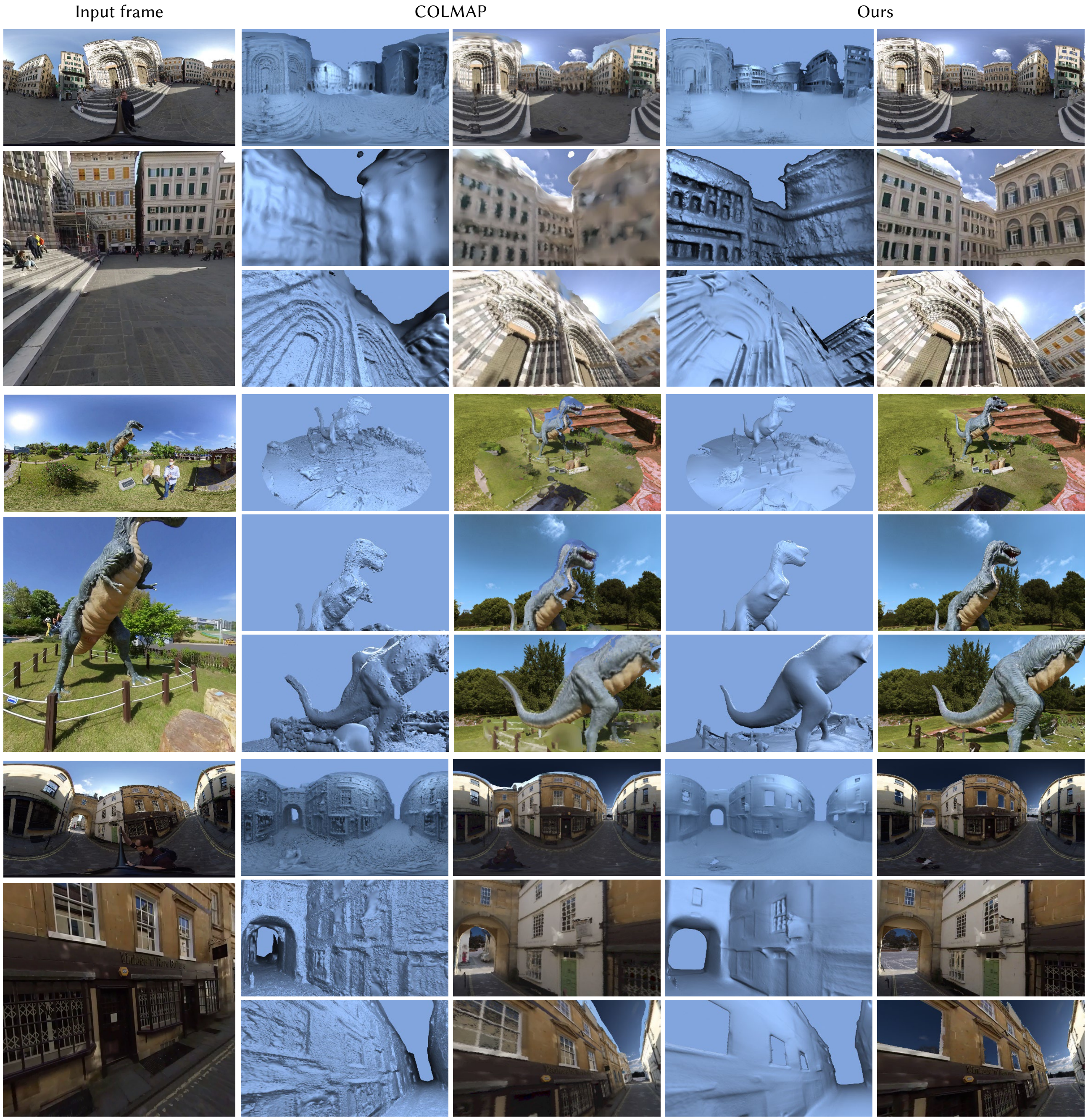


Fig. 10. Comparison of our 3D reconstruction to COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016]. Left: Input frame as spherical image (top) and perspective crop (bottom). Center: COLMAP's reconstructed mesh and textured mesh. Right: Our reconstructed and textured mesh. The top and bottom input videos are from Bertel et al. [2020]. See our supplemental video and document for additional results and comparisons.

4.2 Memory Efficiency

For effective 3D reconstruction, we devised an adaptive data structure with non-uniform voxel sizes, which is critically different from conventional methods. Note that existing voxel-based reconstruction methods allocate a uniform voxel size throughout the entire grid. This means small, less reliable observations of a few pixels could back-project to a large surface far from the camera. This occupies a large number of voxels of uniform size, requiring excessive memory, especially considering the observation's resolution. In Table 2, we quantitatively compare our spherical binoctree with a dense voxel grid, a TSDF-based reconstruction [Nießner et al. 2013], and two types of octree by converting our spherical input to six perspective

Table 2. Given the same image and depth sequence as input, our spherical binoctree is more memory-efficient thanks to adaptively sized voxels. 'Cartesian octree (naïve)' divides voxels always until the final level, while 'Cartesian octree (solid angle)' stops dividing voxels with the same rule as our spherical binoctree. The right-most column shows the volume of voxels.

Scene	Method	# Voxels	GB	Volume [mm^3]
Sponza	Dense regular voxel grid	732,721,016	5.46	1,000
	VoxelHashing [2013]	23,619,456	0.18	1,000
	Cartesian octree (naïve)	110,324,676	2.05	477
	Cartesian octree (solid angle)	14,473,670	0.27	$60 - 5.1{\times}10^{11}$
	Ours	**4,273,474**	**0.08**	$\mathbf{25 - 4.8{\times}10^{12}}$
Cathedral	Dense regular voxel grid	80,762,334	0.60	1,000
	VoxelHashing [2013]	15,561,728	**0.12**	1,000
	Cartesian octree (naïve)	88,189,529	1.64	477
	Cartesian octree (solid angle)	16,035,307	0.30	$0.93 - 5.1{\times}10^{11}$
	Ours	**8,145,273**	0.15	$\mathbf{0.35 - 5.0{\times}10^{12}}$

RGBD images (cubemap, using our poses). Note that all baselines need far more voxels even though their voxels are much larger than our smallest. Matching the level of detail for close surfaces that our method provides would require even more voxels, which would exceed memory capacity.

4.3 Reconstruction Accuracy

We compare our method with two relevant state-of-the-art 3D reconstruction methods: COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016] and OmniSLAM [Won et al. 2020]. Even though these two methods allow for acquiring environment geometry, they are not originally designed to take spherical images as input, unlike our method that reconstructs a full 3D geometry model from spherical inputs. To enable a quantitative comparison, we simulate each method's expected input by rendering 3D scenes and use our estimated camera poses. Since COLMAP takes perspective camera input only, we rendered cube maps with overlapping 120° fields of view. For OmniSLAM, we simulate their camera rig: four 220° fisheye cameras arranged in the corners of a 30 cm square. The center of OmniSLAM's rig followed the same trajectory as the other renderings.

Using the textured meshes produced by each method, we render equirectangular color images and depth maps at the center of the camera path. Figure 9 compares these 3D reconstruction results, color images and depth maps. We compare the results to the ground truth on all mesh pixels, and also on the mesh rendered on top of the original skybox. OmniSLAM's tracking-based pose estimation makes it more robust than COLMAP in this regard, although its reconstruction only covers a limited subset of the vertical field of view. As Table 3 also shows, our method can accurately reconstruct the visible geometry while also correctly estimating the texture. In addition, the low error when estimating depth from our mesh validates that our spherical binoctree effectively integrates depth information across multiple frames.

4.4 Reconstruction of Real Scenes

We compare the reconstruction performance of our method on real scenes with COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016] in Figure 10. For both methods, we use the same input videos captured with an omnidirectional video camera, or converted to cube maps for COLMAP. COLMAP's reconstruction often contains geometric artifacts caused by Poisson surface reconstruction [Kazhdan and Hoppe 2013], and the reconstructed texture information often suffers from blurriness, due to the inaccurate geometry reconstructed by COLMAP. In contrast, our method reconstructs not only high-resolution geometry but also clear texture information without losing details, thanks to our high-accuracy geometry reconstruction. Please see our supplemental video for more results.

Table 3. Textured mesh reconstruction comparison. For each method, we render images and inverse depth maps using the reconstructed mesh. We evaluate the quality of just the mesh pixels ('Mesh') and all pixels ('Mesh+Skybox'). Completeness ('Comp.') is defined as the proportion of pixels that see the mesh compared to the ground truth. Our textured mesh shows the highest geometry and texture accuracy while retaining a high completeness. The large Cartesian octree mesh crashes the texturing pipeline (out-of-memory). See Figure 9 for visual results.

	Depth				Color				Comp.
	Mesh only		Mesh+Skybox		Mesh	Mesh+Skybox			
Methods	MAE	RMSE	MAE	RMSE	PSNR	PSNR	SSIM	LPIPS	%
VoxelHashing	0.025	0.071	0.025	0.070	20.26	21.46	0.840	0.408	97.3
Cartesian octree	0.010	0.024	**0.009**	**0.024**	—	—	—	—	**100**
COLMAP MVS	0.012	0.037	0.011	0.036	22.42	**27.16**	0.892	0.326	**100**
OmniSLAM	0.021	0.046	0.134	0.213	13.52	7.74	0.503	0.730	44.9
Ours	**0.006**	**0.018**	**0.009**	0.038	**23.91**	24.19	**0.922**	**0.142**	98.3

5 DISCUSSION

Our proposed approach focuses on short egocentric camera trajectories, but is also applicable for larger camera trajectories. The binoctree will be divided more finely for close surfaces, and locally behaves similar to other methods such as Nießner et al. [2013] while maintaining large voxels for surfaces that were never observed closely. However, the reconstruction quality might degrade for videos captured with a large camera trajectory because of the increase of depth error caused by photometric inconsistencies, including reflections. Our method is not free from limitations as follows.

Dense Depth Maps. Dense depth maps are necessary to reconstruct 3D scenes from passive multi-view stereo input [Schönberger et al. 2016]. While traditional reconstruction methods depend on feature-based correspondence matching and propagation of sparse depth values, we directly estimate dense depth maps from spherical disparity estimates. We found that this approach can increase depth errors near sharp edges or thin structures in a scene, resulting in the loss of geometric detail (for example, see Figure 12, left).

Consistent Depth. Luo et al. [2020] introduce a method for estimating consistent per-frame depth maps from a video. They combine a spatial color difference loss and disparity loss for fine-tuning a depth estimation network at test time. This approach is preferable to naïve per-frame depth estimation as it can provide depth maps without flickering effects caused by geometrical depth inconsistency. We attempted to adopt this approach in our depth estimation method by replacing the depth estimation network with an optical flow estimation network, and by extending the loss functions to the spherical domain. However, we did not observe a consistent increase in the

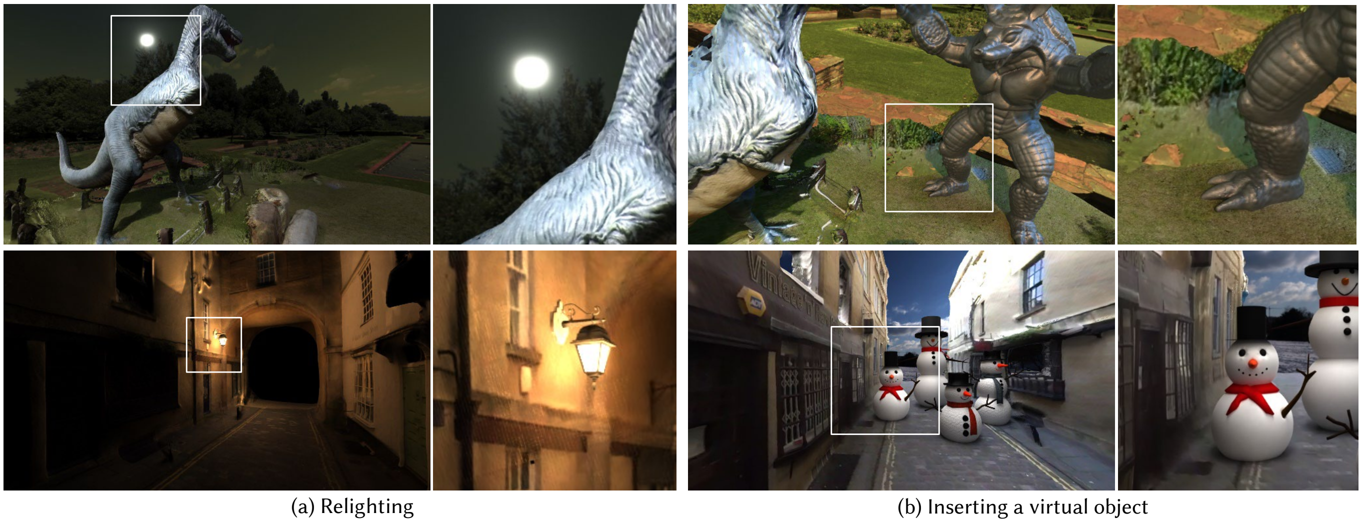


Fig. 11. (a) Day-to-night relighting of captured scenes using virtual point lights. (b) Inserting virtual objects in the scenes.

accuracy of estimated depth. We hypothesize that this is because the structure of our optical flow network did not fit with the losses.

Reflective Surfaces. Our method estimates depth from passive stereo. When a scene includes reflective surfaces, such as windows and mirrors, our depth estimation method tends to produce inconsistent and inaccurate depth maps. Our reconstruction algorithm truncates these depth values, and thus it is hard to reconstruct the geometry of reflective surfaces, as shown in Figure 12 (center).

Dynamic Objects. Our confidence-based TSDF integration enables us to eliminate depth values of moving objects in the scene. We then search the best frame candidates from the input video independently to create a texture atlas. However, our texture mapping applied on a scene with dynamic objects presents ghosting artifacts, because it cannot find the correct texture source through dynamically changing visibility. See Figure 12 (right) for example artifacts. More advanced texture mapping would be interesting future work.

6 CONCLUSION

We have proposed an egocentric 3D reconstruction method, allowing for high-quality geometry and textures even from a short handheld spherical video captured with a relatively small trajectory compared to the scene. We fuse all depth maps using a custom-designed spherical binoctree data structure that divides nodes in a binary or octree fashion to optimally represent egocentric scene geometry. We update TSDF values weighted based on proximity, color and depth consistency, and subject to a depth-dependent truncation threshold. We have demonstrated that the high quality of our reconstructed 3D geometry and textures improve on the state of the art both quantitatively and qualitatively. This enables a range of applications, including novel-view synthesis. We further demonstrate editing of the scene illumination and inserting virtual objects seamlessly in the captured scene in Figure 11. For example, various point light sources are added for the left scene, and virtual objects are inserted on the right. Please see our supplemental video for these and more results.

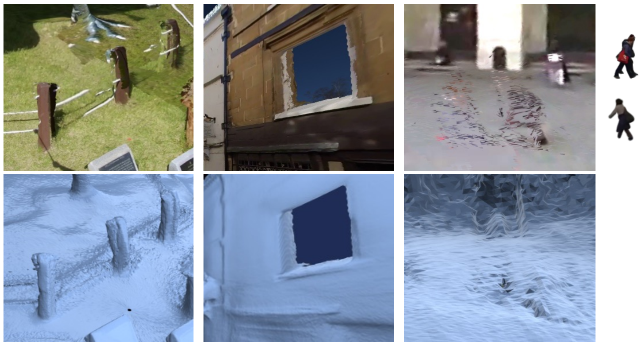


Fig. 12. Example failure cases. Left: Thin objects are hard to reconstruct due to the lack of depth accuracy. Center: Specular reflections can cause incorrect depth estimation and mesh reconstruction. Right: Texture reconstruction can fail for dynamic objects even if mesh reconstruction succeeds.

ACKNOWLEDGMENTS

We thank the reviewers for their valuable feedback that has helped to improve our paper. Min H. Kim acknowledges the MSIT/IITP of Korea (RS-2022-00155620 and 2017-0-00072) and the Samsung Research Funding Center (SRFC-IT2001-04) for developing partial 3D imaging algorithms, in addition to the support of the NIRCH of Korea (2021A02P02-001), Samsung Electronics, and Microsoft Research Asia. Christian Richardt acknowledges an EPSRC-UKRI Innovation Fellowship (EP/S001050/1) and RCUK grant CAMERA (EP/M023281/1, EP/T022523/1).

REFERENCES

Tobias Bertel, Mingze Yuan, Reuben Lindroos, and Christian Richardt. 2020. OmniPhotos: Casual 360° VR Photography. *ACM Trans. Graph.* 39, 6 (2020), 267:1–12. DOI: 10.1145/3414685.3417770

Blender Online Community. 2022. *Blender – a 3D modelling and rendering package.* Blender Foundation. https://www.blender.org/

Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *SIGGRAPH*. 303–312. DOI: 10.1145/237170.237269

Marc Eder, Pierre Moulon, and Li Guan. 2019. Pano Popups: Indoor 3D Reconstruction with a Plane-Aware Network. In *3DV*. 76–84. DOI: 10.1109/3DV.2019.00018

Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. 2017. Casual 3D Photography. *ACM Trans. Graph.* 36, 6 (2017), 234:1–15. doi: 10.1145/3130800.3130828

Peter Hedman and Johannes Kopf. 2018. Instant 3D Photography. *ACM Trans. Graph.* 37, 4 (2018), 101:1–12. doi: 10.1145/3197517.3201384

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable Inside-Out Image-Based Rendering. *ACM Trans. Graph.* 35, 6 (2016), 231:1–11. doi: 10.1145/2980179.2982420

Daniel Hernandez-Juarez, Alejandro Chacón, Antonio Espinosa, David Vázquez, Juan Carlos Moure, and Antonio M. López. 2016. Embedded Real-time Stereo Estimation via Semi-Global Matching on the GPU. In *International Conference on Computational Science*. 143–153. doi: 10.1016/j.procs.2016.05.305

Heiko Hirschmüller. 2008. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Trans. Pattern Anal.* 30, 2 (2008), 328–341. doi: 10.1109/TPAMI.2007.1166

Sunghoon Im, Hyowon Ha, François Rameau, Hae-Gon Jeon, Gyeongmin Choe, and In So Kweon. 2016. All-around Depth from Small Motion with A Spherical Panoramic Camera. In *ECCV*. doi: 10.1007/978-3-319-46487-9_10

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. In *UIST*. 559–568. doi: 10.1145/2047196.2047270

Hualie Jiang, Zhe Sheng, Siyu Zhu, Zilong Dong, and Rui Huang. 2021. UniFuse: Unidirectional Fusion for 360° Panorama Depth Estimation. *IEEE Robotics and Automation Letters* 6, 2 (2021), 1519–1526. doi: 10.1109/LRA.2021.3058957

Lei Jin, Yanyu Xu, Jia Zheng, Junfei Zhang, Rui Tang, Shugong Xu, Jingyi Yu, and Shenghua Gao. 2020. Geometric Structure Based and Regularized Depth Estimation From 360 Indoor Imagery. In *CVPR*. 886–895. doi: 10.1109/CVPR42600.2020.00097

Sing Bing Kang and Richard Szeliski. 1997. 3-D Scene Data Recovery Using Omnidirectional Multibaseline Stereo. *Int. J. Comput. Vis.* 25, 2 (1997), 167–183. doi: 10.1023/A:1007971901577

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* 32, 3 (2013), 29:1–13. doi: 10.1145/2487228.2487237

Hansung Kim and Adrian Hilton. 2013. 3D Scene Reconstruction from Multiple Spherical Stereo Pairs. *Int. J. Comput. Vis.* 104, 1 (2013), 94–116. doi: 10.1007/s11263-013-0616-1

Ren Komatsu, Hiromitsu Fujii, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. 2020. 360° Depth Estimation from Multiple Fisheye Images with Origami Crown Representation of Icosahedron. In *IROS*. doi: 10.1109/IROS45743.2020.9340981

Tilman Kühner and Julius Kümmerle. 2020. Large-Scale Volumetric Scene Reconstruction using LiDAR. In *ICRA*. 6261–6267. doi: 10.1109/ICRA40945.2020.9197388

Po Kong Lai, Shuang Xie, Jochen Lang, and Robert Laganière. 2019. Real-time panoramic depth maps from omni-directional stereo images for 6 DoF videos in virtual reality. In *IEEE VR*. 405–412. doi: 10.1109/VR.2019.8798016

Joo Ho Lee, Hyunho Ha, Yue Dong, Xin Tong, and Min H. Kim. 2020. TextureFusion: High-Quality Texture Acquisition for Real-Time RGB-D Scanning. In *CVPR*. 1272–1280. doi: 10.1109/CVPR42600.2020.00135

Shigang Li. 2008. Binocular Spherical Stereo. *IEEE Transactions on Intelligent Transportation Systems* 9, 4 (2008), 589–600. doi: 10.1109/TITS.2008.2006736

Vadim Litvinov and Maxime Lhuillier. 2013. Incremental Solid Modeling from Sparse and Omnidirectional Structure-from-Motion Data. In *BMVC*.

William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* 21, 4 (1987), 163–169. doi: 10.1145/37401.37422

Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. 2020. Consistent Video Depth Estimation. *ACM Trans. Graph.* 39, 4 (2020), 71:1–13. doi: 10.1145/3386569.3392377

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. Graph.* 21, 3 (2002), 362–371. doi: 10.1145/566654.566590

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 (2017), 148:1–12. doi: 10.1145/3072959.3073645

Morgan McGuire. 2017. Computer Graphics Archive. https://casual-effects.com/data

Andréas Meuleman, Hyeonjoong Jang, Daniel S. Jeon, and Min H. Kim. 2021. Real-Time Sphere Sweeping Stereo from Multiview Fisheye Images. In *CVPR*. doi: 10.1109/CVPR46437.2021.01126

Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. 2016. OpenMVG: Open multiple view geometry. In *International Workshop on Reproducible Research in Pattern Recognition*. 60–74. doi: 10.1007/978-3-319-56414-2_5

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. 2013. Real-time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Trans. Graph.* 32, 6 (2013), 169:1–11. doi: 10.1145/2508363.2508374

Ryan Styles Overbeck, Daniel Erickson, Daniel Evangelakos, Matt Pharr, and Paul Debevec. 2018. A System for Acquiring, Compressing, and Rendering Panoramic Light Field Stills for Virtual Reality. *ACM Trans. Graph.* 37, 6 (2018), 197:1–15. doi: 10.1145/3272127.3275031

Albert Parra Pozo, Michael Toksvig, Terry Filiba Schrager, Joyse 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 (2019), 216:1–16. doi: 10.1145/3355089.3356555

Giovanni Pintore, Marco Agus, Eva Almansa, Jens Schneider, and Enrico Gobbetti. 2021. SliceNet: Deep Dense Depth Estimation From a Single Indoor Panorama Using a Slice-Based Representation. In *CVPR*. 11531–11540. doi: 10.1109/CVPR46437.2021.01137

Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. 2004. Visual Modeling with a Hand-Held Camera. *Int. J. Comput. Vis.* 59, 3 (2004), 207–232. doi: 10.1023/B:VISI.0000025798.50602.3a

Pedro V. Sander, Steven J. Gortler, John Snyder, and Hugues Hoppe. 2002. Signal-Specialized Parametrization. In *Eurographics Workshop on Rendering*. 87–98.

Scott Schaefer and Joe Warren. 2005. Dual Marching Cubes: Primal Contouring of Dual Grids. *Comput. Graph. Forum* 24, 2 (2005), 195–201. doi: 10.1111/j.1467-8659.2005.00843.x

Johannes L. Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *CVPR*. 4104–4113. doi: 10.1109/CVPR.2016.445

Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV*. 501–518. doi: 10.1007/978-3-319-46487-9_31

Ana Serrano, Incheol Kim, Zhili Chen, Stephen DiVerdi, Diego Gutierrez, Aaron Hertzmann, and Belen Masia. 2019. Motion parallax for 360° RGBD video. *IEEE Trans. Vis. Comput. Graph.* 25, 5 (2019), 1817–1827. doi: 10.1109/TVCG.2019.2898757

Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. 2019. OpenVSLAM: a Versatile Visual SLAM Framework. In *International Conference on Multimedia*. doi: 10.1145/3343031.3350539

Cheng Sun, Min Sun, and Hwann-Tzong Chen. 2021. HoHoNet: 360 Indoor Holistic Understanding with Latent Horizontal Features. In *CVPR*. 2573–2582. doi: 10.1109/CVPR46437.2021.00260

Zachary Teed and Jia Deng. 2020. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In *ECCV*. doi: 10.1007/978-3-030-58536-5_24

Fu-En Wang, Hou-Ning Hu, Hsien-Tzu Cheng, Juan-Ting Lin, Shang-Ta Yang, Meng-Li Shih, Hung-Kuo Chu, and Min Sun. 2018. Self-Supervised Learning of Depth and Camera Motion from 360° Videos. In *ACCV*. doi: 10.1007/978-3-030-20873-8_4

Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. 2020b. BiFuse: Monocular 360 Depth Estimation via Bi-Projection Fusion. In *CVPR*. 462–471. doi: 10.1109/CVPR42600.2020.00054

Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 2020a. 360SD-Net: 360° Stereo Depth Estimation with Learnable Cost Volume. In *ICRA*. 582–588. doi: 10.1109/ICRA40945.2020.9196975

Katja Wolff, Changil Kim, Henning Zimmer, Christopher Schroers, Mario Botsch, Olga Sorkine-Hornung, and Alexander Sorkine-Hornung. 2016. Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction. In *3DV*. 118–127. doi: 10.1109/3DV.2016.20

Changhee Won, Jongbin Ryu, and Jongwoo Lim. 2019a. OmniMVS: End-to-End Learning for Omnidirectional Stereo Matching. In *ICCV*. 8986–8995. doi: 10.1109/ICCV.2019.00908

Changhee Won, Jongbin Ryu, and Jongwoo Lim. 2019b. SweepNet: Wide-baseline Omnidirectional Depth Estimation. In *ICRA*. doi: 10.1109/ICRA.2019.8793823

Changhee Won, Hochang Seok, Zhaopeng Cui, Marc Pollefeys, and Jongwoo Lim. 2020. OmniSLAM: Omnidirectional Localization and Dense Mapping for Wide-baseline Multi-camera Systems. In *ICRA*. 559–566. doi: 10.1109/ICRA40945.2020.9196695

Ming Zeng, Fukai Zhao, Jiaxiang Zheng, and Xinguo Liu. 2013. Octree-based fusion for realtime 3D reconstruction. *Graphical Models* 75, 3 (2013), 126–136. doi: 10.1016/j.gmod.2012.09.002

Wei Zeng, Sezer Karaoglu, and Theo Gevers. 2020. Joint 3D Layout and Depth Prediction from a Single Indoor Panorama Image. In *ECCV*. doi: 10.1007/978-3-030-58517-4_39

Jianing Zhang, Tianyi Zhu, Anke Zhang, Xiaoyun Yuan, Zihan Wang, Sebastian Beetschen, Lan Xu, Xing Lin, Qionghai Dai, and Lu Fang. 2020. Multiscale-VR: Multiscale Gigapixel 3D Panoramic Videography for Virtual Reality. In *ICCP*. doi: 10.1109/ICCP48838.2020.9105244

Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. 2004. Iso-Charts: Stretch-Driven Mesh Parameterization Using Spectral Analysis. In *Symposium on Geometry Processing (SGP)*. 45–54. doi: 10.1145/1057432.1057439

Qian-Yi Zhou and Vladlen Koltun. 2014. Color Map Optimization for 3D Reconstruction with Consumer Depth Cameras. *ACM Trans. Graph.* 33, 4 (2014), 155:1–10. doi: 10.1145/2601097.2601134

Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, Federico Alvarez, and Petros Daras. 2019. Spherical View Synthesis for Self-Supervised 360° Depth Estimation. In *3DV*. 690–699. doi: 10.1109/3DV.2019.00081

Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. 2018. OmniDepth: Dense Depth Estimation for Indoors Spherical Panoramas. In *ECCV*. 448–465. doi: 10.1007/978-3-030-01231-1_28