Real-Time Sphere Sweeping Stereo from Multiview Fisheye Images

Andreas Meuleman Hyeonjoong Jang Daniel S. Jeon Min H. Kim

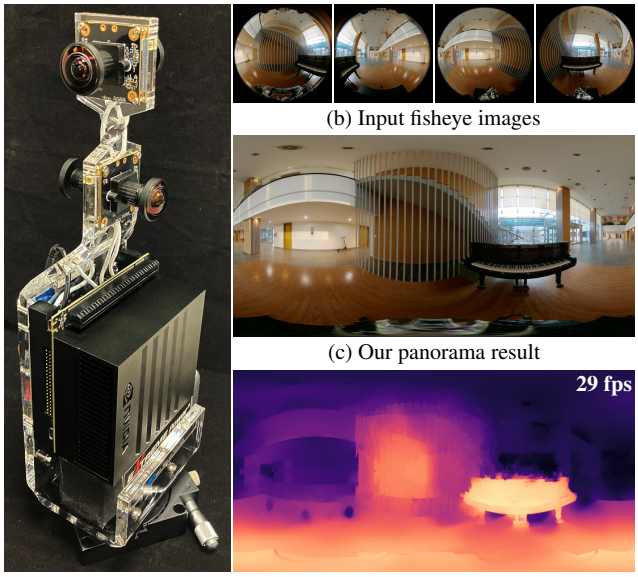
KAIST

Abstract

A set of cameras with fisheye lenses have been used to capture a wide field of view. The traditional scan-line stereo algorithms based on epipolar geometry are directly inapplicable to this non-pinhole camera setup due to optical characteristics of fisheye lenses; hence, existing complete 360° RGB-D imaging systems have rarely achieved real-time performance yet. In this paper, we introduce an efficient *sphere-sweeping stereo that can run directly on multiview fisheye images without requiring additional spherical rectification. Our main contributions are: First, we introduce an adaptive spherical matching method that accounts for each input fisheye camera's resolving power concerning spherical distortion. Second, we propose a fast inter-scale bilateral cost volume filtering method that refines distance in noisy and textureless regions with optimal complexity of $O(n)$. It enables real-time dense distance estimation while preserving edges. Lastly, the fisheye color and distance images are seamlessly combined into a complete 360° RGB-D image via fast inpainting of the dense distance map. We demonstrate an embedded 360° RGB-D imaging prototype composed of a mobile GPU and four fisheye cameras. Our prototype is capable of capturing complete 360° RGB-D videos with a resolution of two megapixels at 29 fps. Results demonstrate that our real-time method outperforms traditional omnidirectional stereo and learning-based omnidirectional stereo in terms of accuracy and performance.*

1. Introduction

Efficient and accurate understanding of the appearance and structure of 3D scenes is a vital capability of computer vision used in many applications, such as autonomous vehicle [38], robotics [60], augmented/mixed reality [51, 5], etc. Conventional stereo cameras with ordinary lenses provide a narrow field of view, insufficient to capture scenes in all directions. In order to capture scenes in all directions, we can build a multi-camera setup like a light-field camera array [6, 44], but it significantly increases manufacturing cost, in addition to computational cost for processing multiple input, to obtain omnidirectional panorama and distance.



(a) Our prototype (b) Input fisheye images (c) Our panorama result (d) Our distance result

Figure 1: (a) Our prototype built on an embedded system. (b) Four input fisheye images. (c) & (d) Our results of omnidirectional panorama and dense distance map (shown as the inverse of distance). It took just 34 ms per frame on this device. Refer to the supplemental video for real-time demo.

It is a natural choice to use a smaller number of fisheye lenses to reduce the number of cameras while covering all directions. The omnidirectional camera configuration with multiple fisheye lenses suffers from an inevitable tradeoff between performance and accuracy when computing full 360° panorama and distance due to the optical characteristics of fisheye lenses presented subsequently.

First, the conventional pinhole camera model is invalid for field of views of 180° or more even when the calibration model can accommodate a wider FoV [25, 12, 53]. Accordingly, unlike ordinary stereo, we cannot find stereo correspondence rapidly by sweeping plane candidates [24] in wide angle fisheye images. Second, epipolar lines on fisheye images are curved [39, 50], requiring warp-aware correspondence search with spatial variation, significantly increasing computational costs. An equirectangular or a latitude-longitude projection can be employed to obtain straight epipolar lines [52, 34]. However, it introduces se-

vere image distortion, and a given disparity does not correspond to the same distance depending on its position in the image. Simply estimating the disparity in the equirectangular domain before converting to distance [32, 35] breaks the local disparity consistency assumption of cost aggregation. Lastly, we cannot merge multiview fisheye images as a 360° panorama image accurately without having a 360° *dense* distance map, and a clear 360° dense distance map cannot be filtered and obtained without a 360° panorama image. It is a chicken-and-egg problem when combining multiview fisheye images to a 360° RGB-D image with high accuracy.

In this work, we propose *real-time* sphere-sweeping stereo that can run directly on multiview fisheye images, without requiring additional spherical rectification using equirectangular or latitude-longitude projection by tackling three key points. First, we propose an adaptive spherical matching method that allows us to evaluate stereo matching directly on the fisheye image domain with consideration of the regional discrimination power of distance in each fisheye image. Second, we introduce fast inter-scale cost volume filtering of optimal complexity $O(n)$ that allows for a stable sphere sweeping volume in noisy and textureless regions. It enables 360° *dense* distance estimation in all directions in real time while preserving edges. Lastly, colors at different distance maps are combined into a *complete* 360° panorama and distance map seamlessly through fast inpainting using the dense distance map.

We implemented our algorithm on a prototype made of an embedded computer with a mobile GPU and four fisheye cameras (Figure 1). Our prototype captures complete 360° RGB-D video that includes color and distance at every pixel with a resolution of two megapixels at 29 fps. Results validate that our real-time algorithm outperforms the traditional omnidirectional stereo and the learning-based 360° stereo algorithms in terms of accuracy and performance.

2. Related Work

Binocular Fisheye/360° Stereo. Two fisheye cameras [31, 32, 30, 13, 45, 46] or 360° cameras [2, 48, 35, 56] are placed on a baseline and then are used to estimate depth (more accurately distance in omnidirectional stereo) within the stereo field of view. Analogue to the traditional epipolar geometry, they apply spherical rectification and matching along the great circle. However, disparity in spherical stereo is proportional to the length of arc, which is not linearly proportional to the inverse of distance, hence requiring exhaustive correspondence search. An equirectangular or a latitude-longitude projection has been commonly used to rectify fisheye images before stereo matching [31, 32, 30, 35, 56, 28]. This process causes severe image distortion and disturbs correspondence search. Also, in this binocular setup, distance cannot be estimated properly along the baseline axis [35, 56], i.e., no complete 360° panorama and distance maps can be computed directly from this binocular stereo setup due to occlusion between the cameras and, most importantly, due to the absence of exploitable baseline in the alignment. Our method uses just four cameras (same number as binocular 360° stereo methods [35, 56]) with fisheye lenses, but it can capture *complete* 360° RGB-D videos in real time. Note that methods that determine the relative position of the scene with planar depth estimation [22] are inherently limited to FoVs below 180°.

Monocular 360° Stereo. The traditional structure-from-motion algorithm has been applied to compact 360° imaging [7, 23, 42]. However, these methods assume that a 360° camera moves in static scenes. If these methods are applied to a scene with dynamic objects, their performances degrade rapidly. Also, computational costs of these methods are expensive, so they are inapplicable to *real-time* 360° RGB-D imaging. In addition, monocular stereo imaging has been applied to 360° panoramas by learning an omnidirectional image prior [62, 55]. Learned priors help matching correspondences in warped images. However, owing to the model complexity, no real-time learning-based method exists yet. Also, to date, there is no real-world dataset of omnidirectional RGB-D images available for deep learning. These methods have been trained on synthetically rendered images of hand-made 3D models and 3D scanning [8, 3, 54, 62, 59, 56, 28]. Owing to the domain gap between real and rendered images, these models often present suboptimal performance with unseen real-world data.

Multiview Fisheye Stereo. Multiple fisheye cameras have been combined to capture 360° RGB-D images with a number of cameras ranging from 4 to 20 [33, 1, 11, 6, 44]. When the camera count increases, the quality of color and distance images is improved significantly, but with rapid increase in hardware and computation cost. When combining multiview fisheye stereo images, technical challenges still exist hindering the real-time performance of this setup. First, for accounting for reprojection, occlusion and visibility of distance values in the unified omnidirectional image space, we need a complete 360° guide image, which cannot be obtained from multiview input without a dense 360° distance map. Only simple warp and blending methods were proposed without distance awareness [21, 20]. As they are designed for a short baseline, they often suffer from stitching artifacts when disparity changes in the overlapping regions. Second, due to the geometry of 360 matching, multiple true matches may occur. This has been handled by devising a computationally intensive cost aggregation [58, 26].

In contrast, we use the minimal number of fisheye cameras to cover *complete* 360° angles in real time so that we keep the manufacturing cost and computational requirement as low as possible. We propose an effective camera design and an adaptive spherical matching algorithm at multiple scales to handle the aforementioned challenges.

3. Fast Sphere Sweeping Stereo

Hardware Design. Our hardware setup employs a minimum number of cameras to achieve a 360° RGB-D image, i.e., four cameras with fisheye lenses. Each fisheye camera has the field of view of 220°. A pair of front-backward fisheye cameras is placed on the top, and another pair of fisheye cameras is placed on the bottom but in a perpendicular direction (Figure 1), so that each combination of neighboring stereo pairs has the same baseline.

Spherical Geometry. We base our work on the classical binocular stereo model [30]. Each pixel in the reference frame I_{c_0} captured by reference camera c_0 describes the perceived color of a ray at angle of polar coordinates (θ, ϕ). It corresponds to the point of polar coordinates (θ, ϕ, d), where d is a distance. This leads to the following 3D position p in c_0's space: $p = d\left[\sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)\right]^\intercal$.

Suppose another camera c_1 at rotation R_{c_1} and position T_{c_1} w.r.t. the reference camera c_0. The camera captures the images I_{c_0} and I_{c_1}. Position p in c_1's space is: $p_{c_1} = R_{c_1}^{-1}(p - T_{c_1})$. Let $\hat{p}_{c_1} = p_{c_1}/\|p_{c_1}\|$ be the normalized vector of p_{c_1}, which is the pixel coordinates in I_{c_1}: $(\theta_{c_1}, \phi_{c_1}) = \left(\arccos(\hat{p}_{c_1\,z}), \frac{3\pi}{2} - \arctan2(\hat{p}_{c_1\,y}, \hat{p}_{c_1\,x})\right)$. The pixel coordinates in c_1 with camera transformation $R_{c_1}|T_{c_1}$ can be expressed as the projection of pixel of angle (θ, ϕ) at distance d in the reference coordinate system: $(\theta_{c_1}, \phi_{c_1}) = \bar{P}_{R_{c_1}|T_{c_1}}(\theta, \phi, d)$. Assuming Lambertian surfaces in a scene, a pixel $I_{c_1}(\theta_{c_1}, \phi_{c_1})$ of camera c_1's image is the same as $I_{c_0}(\theta, \phi)$ in the reference camera. Pixels in other cameras' images can be expressed w.r.t. the reference coordinate system in the same way.

Sphere Sweep Volume. Similarly to multi-view stereo with standard camera model, we build a sweep volume for several distance candidates $d_0, ..., d_{N-1}$. Instead of warping I_{c_1} to I_{c_0} following homographies with planar distance candidates [24], we use the previously described mapping and spheres distance candidates around the reference frame or a given point [59].

For each candidate, we create a warped version of I_{c_1} that can be matched to I_{c_0} if the distance candidate is correct. In details, for all pixels coordinates (θ, ϕ) in I and for all distance candidate d_i, we find the corresponding coordinates. This process is depicted in Figure 2. We then assign the values of the sphere sweep volume V as follows: $V_{c_1 \to c_0}(\theta, \phi, i) = I_{c_1}(\bar{P}_{R_{c_1}|T_{c_1}}(\theta, \phi, d_i))$. In this volume, a correct distance candidate d_k shows good matching: $V_{c_1 \to c_0}(\theta, \phi, k) \approx I_{c_0}(\theta, \phi)$, which provides a direct cue for distance estimation. The quality of matching can be evaluated through photometric difference or difference after image transformation: gradient [17], census transform [29] or feature extraction [59]. For robust performance, cost aggregation [19, 18, 61] or deep regularization [57] is necessary when selecting the best depth candidate. Several views can also be used simultaneously [24].

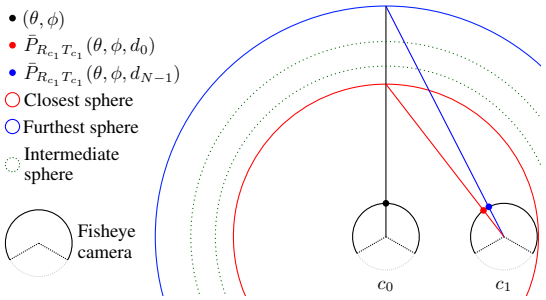


Figure 2: Projection in the sphere sweep volume. A pixel of coordinates (θ, ϕ) in camera c_0 has different coordinates in c_1 depending on the scene's distance: the red and the blue points correspond to pixel positions for two extreme distances. Circles red and blue represent the closest and furthest distance candidates around c_0 while the doted ones are intermediate candidates in the sweeping volume.

3.1. Adaptive Spherical Matching

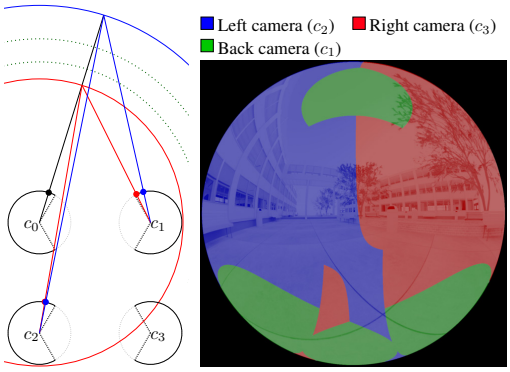
In theory, we can evaluate entire depth candidates in all possible combinations of overlapped regions along the baseline in the sphere sweeping volume. It is exhaustive computation. For achieving real-time performance, we tailor a camera selection method that provides the regional best camera pairs for search correspondence in the sphere sweeping volume w.r.t. the reference camera.

We select only the best camera among other three cameras (c_1, c_2 and c_3) for each pixel in the reference view. If several cameras have a field of view that covers a pixel in the reference frame, we select the one that has the highest distance discriminating power. This property can be described as maximizing the difference between the layers of the sphere sweep volume and be able to identify which candidate matches the best more clearly.

To quantify this for a given pixel position (θ, ϕ) in the reference image I_{c_0}, we focus on the first and last layers 0 and $N-1$ of the volume, corresponding to the distance candidates d_0 and d_{N-1}. Let $p_{c_k}^{<i>}$ be the point in camera c_k's space of reference coordinates (θ, ϕ, d_i). The best camera c_k is the one that shows the most angular change between two 3D points $p_{c_k}^{<0>}$ and $p_{c_k}^{<N-1>}$ given from these two distance candidates.

In detail, if the angle between $p_{c_k}^{<0>}$ and $p_{c_k}^{<N-1>}$ is high, the sampled location in the selected camera for the sweeping volume will change significantly, which is suitable for distance estimation. We define the discriminating power weight based on those considerations: $q_{c_k} = \left|\arccos\left\langle \hat{p}_{c_k}^{<0>} \cdot \hat{p}_{c_k}^{<N-1>} \right\rangle\right|$, with $\hat{p} = p/\|p\|$ are normalized vectors. Using this evaluation, we select the optimal camera c^* for each pixel in the reference following:

$$c^*(\theta, \phi) = \underset{c_k}{\operatorname{argmax}}(q_{c_k}). \tag{1}$$



(a) Best camera evaluation

(b) Selected camera for each pixel in c_0

Figure 3: (a) We evaluate how a ray in c_0 is reprojected in c_1 and c_2. For this pixel, the best camera for matching is the one that shows the maximum displacement for a given distance: if a small change of distance leads to a high displacement, the distance discriminating power is improved. (a) shows that, for this specific ray in c_0, c_1 is a better camera for matching than c_2, despite the baseline between both pairs being similar. (b) For each pixel in c_0, the camera showing the best distance discrimination is selected. Note that (a) shows a hypothetical layout for visualization.

Figure 3 shows an example of camera selection for a given ray angle in the reference frame and a map of which camera is selected depending on the pixel position.

3.2. Efficient Spherical Cost Aggregation

Once we built the prototype, we calibrated the four cameras using the double sphere model [53]. We perform two 220° distance estimation using the two opposed top cameras as two references. For each pixel in each reference, we select the best camera using our selective matching. Let I_{c_s} be the image from the camera selected at pixel (θ, ϕ) and I_{c_0} be the reference frame. The matching cost for the i_{th} distance candidate is: $C(\theta, \phi, i) = \|V_{c_s \to c_0}(\theta, \phi, i) - I(\theta, \phi)\|_1$, where $V_{c_s \to c_0}$ is the sphere sweeping volume from the selected camera to the reference one. We then regularize each slice of the spherical cost volume using our fast filtering method described in the following section. After cost volume filtering, the optimal distance is voted via winner-takes-all, and sub-candidate accuracy is achieved through quadratic fitting.

3.2.1 Fast Inter-Scale Bilateral Filtering

For aggregating sparse distance to obtain a dense distance map, there are many available methods that smooths costs in an edge-aware manner. Bilateral grid-based methods [4, 10], while showing impressive capabilities, are still computationally expensive to be applied on the 3D cost volume and often produce blocky artifacts even with domain transform post processing [14]. A more hardware-friendly

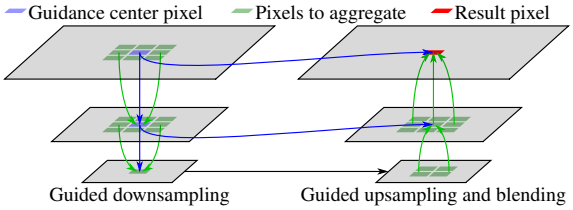


Figure 4: Inter-scale bilateral filtering. We first downsample with edge preservation using the bilateral weights between the guidance center and the neighbor pixels to aggregate sparse costs. Then, we upsample using a minimal pixel support. We use guidance weights computed between the guidance centers and the pixels to aggregate at lower scale.

version of the fast bilateral solver has been devised [36]. While having demonstrated strong performance for a single depth map, it is more hardware specific and is still not computationally efficient enough to be applied to a complete cost volume in real time. Another popular edge-aware filtering is the guided filter [16, 15], which has been used with cost volume pyramids [43] or multi-scale cost aggregation [61]. While showing optimal complexity of $O(n)$, they cannot perform fast on GPU because they suffer from computation overhead when computing integral images in parallel environments. To achieve two megapixels real-time RGB-D imaging at 29 fps on an embedded machine with a GPU, we introduce a fast inter-scale bilateral filtering method specially designed for parallel computing environments.

Edge-Preserving Downsampling. The first step of our filtering is to downscale input image without blurring, thus preventing edge bleeding and halos. To this end, we perform filtering with the neighbor pixels with bilateral weights before decimation. We define I_0 the original image and I_l image after being downsampled by two l times. We first define bilateral weights:

$$w^{\downarrow}_{mn}(I, x, y) = \exp\left(\frac{\|I(x, y) - I(x + m, y + n)\|^2}{2\sigma_I^2}\right), \quad (2)$$

where σ_I is the edge preservation parameter, and (x, y) are pixel coordinates. We define the downsampling scheme as:

$$I^{\downarrow}(x, y) = \sum_{m,n=-1}^{1} I(2x + m, 2y + n) w^{\downarrow}_{mn}(I, 2x, 2y)/\tau, \quad (3)$$

where τ is the normalizing constant. In the pyramid, we note $I_{l+1} = I_l^{\downarrow}$. We define the number of scale levels L.

Edge-Preserving Upsampling. Unlike existing edge-preserving upsampling methods [27, 9] that use the high resolution image as guide, our method uses the bilateral weights between the downsampled and the full resolution image to achieve optimal complexity. Note that we intentionally do not use Gaussian spatial weights proposed by Kopf et al. [27] to focus on efficiency, and as they are designed for a wider support.

In addition to bilateral weights, we blend the scales using a Gaussian function of the current scale index. They are

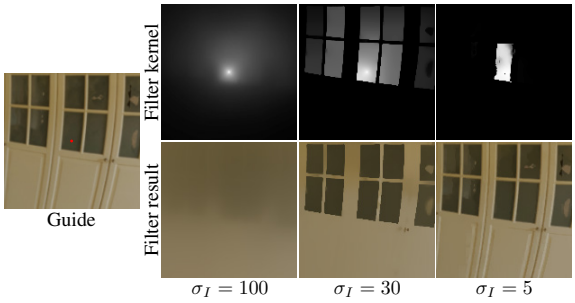


Figure 5: Impact of the filter kernel with different edge preservation parameters. Our method yields high preservation and smoothing capabilities with global support while showing minimal runtime.

defined for each scale as $w_l^{\uparrow} = \exp\left(\frac{(2^l)^2}{2\sigma_s^2}\right)$, where σ_s is the smoothness parameter. The weight for the higher resolution scale is naturally $1 - w_l^{\uparrow}$. Figure 4 depicts our multi-scale filtering process.

Filter Kernel. The final filter kernel obtained after this downsampling/upsampling process yields smooth fall off driven by σ_s as we move away from the center and does not cross edge boundaries as shown in an example in Figure 5. Although the each step of the algorithm only covers a minimal pixel support, the bilateral downsampling/upsampling filtering yields a kernel that covers the *entire* image. The guidance through the bilateral weights is a composition of exponential with a higher order far from a given pixel. This naturally provides increased guidance between spaced pixels. For our results, σ_s is set to 25, and σ_I is set to 10.

Complexity. The number of operations follows the sum of a geometric series with ratio $\frac{1}{4}$. The asymptotic complexity is therefore $O(n)$ with n the number of pixels, making the algorithm optimal. The number of levels has to allow the lowest level L to have a size above one pixel. We therefore downsample at most $\ln_4(n)$ times. While the downsampling and upsampling have to be run sequentially with $O(\ln(n))$ levels, each downsampling and upsampling steps are fully parallelized.

3.3. Distance-aware Panorama Stitching

Instead of estimating distance at the center, our distance estimation algorithm relies on reference frames for edge preservation and to avoid multiple true matches (Figure 6). While this approach yields an increased accuracy, an extra step is required to merge the fisheye images. We present an efficient method that first synthesize a distance map at a desired location, then project the image following the 3D coordinates and finally merges the images through a blending process giving more weight to the least displaced pixels.

Novel View Synthesis. The first step is to reproject the dense distance maps to a selected location, common to both references. To this end, we find for each pixel (θ, ϕ) its corresponding position, and translate them to the selected location and find the coordinates (θ_r, ϕ_r) in the reprojected image. We obtain: $(\theta_r, \phi_r) = \bar{P}_{T^*}(\theta, \phi, \hat{D}(\theta, \phi))$, where T^* is the desired position with respect to the camera and $\hat{D}$ is the estimated distance map. This forward warping operation leads inevitably to multiple pixels in the original distance map mapping to the same target pixel, i.e., several couples (θ, ϕ) may be projected to the same coordinates (θ_r, ϕ_r). This ambiguity requires splatting to obtain the final value.

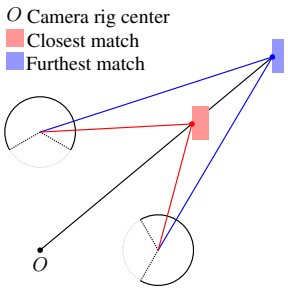


Figure 6: A ray from O crosses multiple objects that are seen from both cameras, meaning that several depth candidates have true matches. We instead stitch fisheye images using depth maps.

We merge the possible pixels in an occlusion aware manner following [40]. Specifically, we use minimum distance splatting, i.e., z-buffering, hence the reprojected distance:

$$\hat{D}_r(\theta_r, \phi_r) = \min \hat{D}(\theta, \phi), \quad \text{s.t. } \bar{P}_{T^*}(\theta, \phi, \hat{D}(\theta, \phi)) = (\theta_r, \phi_r). \tag{4}$$

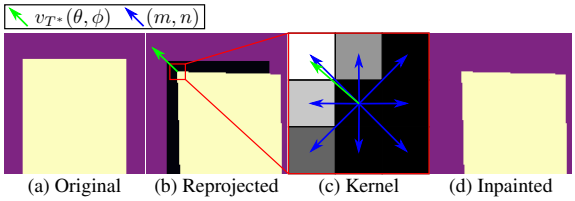
Directional Inpainting. While some pixels in the target can have several counterparts in the original distance map, some pixels have none due to occlusion. We inpaint the missing regions using the background as they can be occluded by the foreground objects. To this end, we first determine the background-to-foreground direction. This is given by the derivative of the projection w.r.t. the distance. Indeed, occlusion holes in the reprojected map are caused by areas with different distance not being reprojected at the same location. We therefore define the inpainting direction: $v_{T^*}(\theta, \phi) = \frac{\partial \bar{P}_{T^*}(\theta, \phi, d)}{\partial d}$. This inpainting direction leads to a directed diffusion kernel that can be used iteratively as proposed by [41]. We determine kernel weights around each pixel depending on their similarity with the inpainting direction: $w_{m,n} = \langle v_{T^*}(\theta, \phi) \cdot (m, n) \rangle^+$, where $^+$ is the positive part and $(m, n) \in [\![-1, 1]\!]^2 \backslash (0, 0)$ are the indices of the eight neighbor pixels. As the dot product gives high weights for aligned vectors, this method naturally creates a diffusion kernel that uses the values of the pixels aligned with the inpainting direction (Figure 8).

Once the distance is moved to the given point of view, we simply project the color pixel following the 3D coordinates given in the distance map, providing the RGB image at a different location.

Blending. After projecting color images to a common location, the two 220° images need to be merged together to create a complete panorama stored in the standard equirectangular projection. To that end, we provide blending weights that correspond to the amount of possible occlu-

Fisheye input Inverse distance Panorama Closeup

Figure 7: Real results captured with our prototype. Refer to the supplemental video for real-time demo of our prototype.



(a) Original (b) Reprojected (c) Kernel (d) Inpainted

Figure 8: Depth at the camera position (a) is projected to the desired view in a depth-aware manner (b). As occlusion generates holes in the projected distance map, we compute an inpainting kernel (c) depending on the occlusion direction. We finally apply this inpainting kernel to the distance map to remove holes using the background depth values (d).

sion. In pixels where $v_{T^*}(\theta,\phi)$ is large, distance changes can modify the image greatly, introducing wider occluded regions, more distance related distortion and potential artifacts. We therefore define the blending weights following a Gaussian on the length of this vector: $b_{c_k}(\theta,\phi) = \exp\left(-\frac{\|v_{T^*}(\theta,\phi)\|_2^2}{2\sigma^2}\right)$. Note that we handle the pixels that cannot be captured by the camera by setting $b_{c_k}(\theta,\phi) = 0$. We estimate the derivatives through finite difference over the distance range.

4. Results

Prototype. We install four cameras using Sony IMX477 sensors on an NVIDIA Jetson Xavier embedded computer with a mobile GPU. The cameras look at four different directions to have both horizontal and vertical baselines (6.8 cm), see Figure 1. We capture and process four fisheye frames of 1216×1216 px and output 2048×1024 px distance maps and panoramas. Two reference frames's resolution is 1024×1024 px. We therefore process a total of 2.1 Mpx. For our tests, we used 32 distance candidates with a [0.55, 100] m range. On the NVIDIA Jetson Xavier, our algorithm computes a RGB-D frame within 34 ms in total, including: cost computation: 14 ms, cost regularization: 10 ms, distance selection and sub-candidate interpolation: 3 ms, and reprojection and stitching: 7 ms. For experiments on a desktop computer, we use an AMD Ryzen Threadripper 3960X with an NVIDIA GTX 1080 Ti. Figure 7 shows results.

Dependency on the camera rig. Our method can be generalized to other rig layouts without modification as long as two cameras cover the entire field of view for stitching. It is also possible to use and stitch more than two references with the same methodology in case no couple of cameras covers the entire field of view. In addition, when using more than four cameras, our adaptive matching will not be prone to a significant change in performance or accuracy.

Synthetic Dataset. To evaluate a wider variety of camera setups with ground truth panorama and distance maps, we build our own rendered dataset. We render 95 frames at random locations with ten different scenes, collected from McGuire computer graphics archive [37] and the official Blender website [49]. Each frame is composed of four fisheye images with 220° FoV. We follow positions and rotation shown in OmniHouse [58] for fair comparison with learning-based methods trained on this dataset, with scaling to match our prototype's form factor. In addition, we render the ground truth RGB panorama and distance maps at the center of the rig as equirectangular images and an additional equirectangular RGB panorama at 7.5 cm to create a stereo pair and evaluate spherical binocular stereo methods. We constrain a minimum distance between the camera rig and the scene of 0.55 m to ensure a distance map within the range of all methods.

Metric. We evaluate the distance quality in its inverse domain on equirectangular images. We define the distance error for one pixel as:

$$E(\theta,\phi) = \left| \frac{1}{\hat{D}(\theta,\phi)} - \frac{1}{D^*(\theta,\phi)} \right|, \quad (5)$$

where $\hat{D}$ is the estimated distance and D^* is the ground

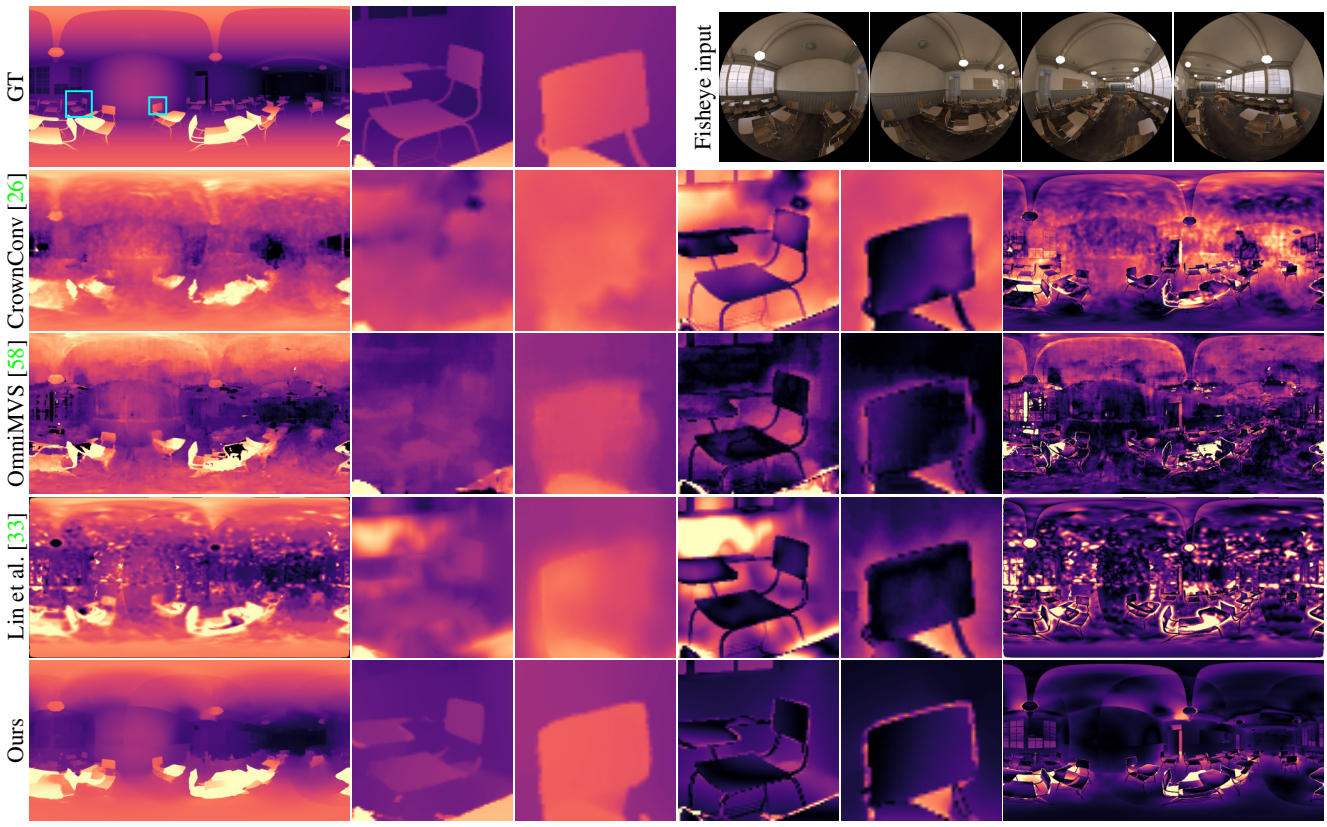


Figure 9: Distance results from fisheye images on our synthetic dataset. Refer to the supplemental document for more results.

truth in meters. We evaluate the proportion of bad pixels, noted $>x$ in our tables, which represents the percentage of pixel with $E(\theta, \phi) > x$ in the output distance map.

Quantitative Evaluation. We analyze the impact of our adaptive matching. Table 1 shows that our selection of the best camera (Equation (1)) marginally increases quality over naïve variance calculation while reducing the runtime. Using all views is not beneficial for the quality as when a camera's baseline is aligned with the reference, its contribution for candidate discrimination is low. Simultaneously, using more cameras yields increased computational cost.

	Inverse distance (% | m^{-1})				Runtime (ms)	
	>0.1	>0.4	MAE	RMSE	1080 Ti	Xavier
All views (naïve)	14.81	0.57	0.056	0.082	$5.7{\cdot}10^{0}$	$2.5{\cdot}10^{1}$
Adaptive matching	**12.51**	**0.55**	**0.053**	**0.079**	$\mathbf{2.7{\cdot}10^{0}}$	$\mathbf{1.4{\cdot}10^{1}}$

Table 1: Impact of our adaptive matching on the distance quality for 2048×1024 px. All views indicates that we used the variance between all cameras during matching, while adaptive matching uses only the best camera per pixel for cost computation. Runtimes only include cost volume computation.

We compare our inter-scale bilateral filter against the fast bilateral solver [4], the guided filter [16], the fast guided filter [15] and cross-scale cost aggregation [61]. When filtering each slice of the cost volume using the fast bilateral solver, we use guide precomputation and OpenMP parallelization. We pair cross-scale stereo with a 7×7 box filter for being the best reported runtime and with the guided filter for showing the best results on the Middlebury dataset [47]. We implement cross-scale stereo, the guided filter and the fast guided filter on GPU for fair runtime comparison. We also precompute guide variance and mean for the guided filters and use an aggressive downsampling ratio $s = 4$ for the fast guided filter. For fair comparison, we tune hyperparameters for all competing methods so that RMSE is minimized. For the fast bilateral, we find $\sigma_{xy} = \sigma_l = \sigma_{uv} = 10$ (optimized together) and $\lambda = 385$. For cross-scale is paired with box, we find $\lambda = 17.5$. We find $\varepsilon = 0.105$ for the guided filter and $\varepsilon = 0.63$ for the fast guided filter.

Table 2 shows comparison between different cost aggregation methods. Our method shows competitive results against much more computationally intensive methods such as the bilateral solver [4]. In addition, adaptation and careful implementation of the bilateral solver [36], although achieving high quality depth super-resolution, reports runtimes that are significantly higher than our method while running on dedicated hardware, thus making it unsuitable for real-time cost volume filtering. The guided filters, due to their local nature, are not able to handle large textureless regions and the hyperparameter tuning tend to lead to weak

	Inverse distance (% | m^{-1})				Runtime (ms)	
	>0.1	>0.4	MAE	RMSE	1080 Ti	Xavier
Fast bilateral solver [4]	12.52	0.58	0.054	0.081	$3.7{\cdot}10^{2}$	$1.4{\cdot}10^{3}$
Guided filter [16]	32.08	8.89	0.136	0.238	$1.4{\cdot}10^{2}$	$2.9{\cdot}10^{2}$
Fast guided filter [15]	34.18	8.34	0.135	0.232	$1.5{\cdot}10^{1}$	$5.0{\cdot}10^{1}$
Cross-scale + Box [61]	17.40	0.93	0.062	0.093	$3.8{\cdot}10^{1}$	$6.4{\cdot}10^{1}$
C-S + G-F [61, 16]	17.39	2.03	0.070	0.115	$2.4{\cdot}10^{2}$	$4.6{\cdot}10^{2}$
Ours	**12.51**	**0.55**	**0.053**	**0.079**	$\mathbf{3.5{\cdot}10^{0}}$	$\mathbf{1.0{\cdot}10^{1}}$

Table 2: Comparison of our cost aggregation on the distance quality for 2048×1024 px with other filtering methods. Note that we implement all methods on GPU except the fast bilateral solver that benefits from a multithreaded implementation. Our method shows competitive accuracy while being orders of magnitude faster.

guidance, with large regularization parameters ε. Cross-scale cost aggregation [61] is able to efficiently handle large textureless regions. However, due to the absence of inter-scale guidance, it is not able to maintain edge preservation even when paired with the guided filter.

We compare distance accuracy of our method against four distance from fisheye methods [33, 59, 58, 26] and three distance from rectified spherical stereo image pairs [31, 35, 56]. As we use their rigs layout, we do not retrain data driven methods. To compare against 360SD-Net [56], we used the pretrained weights that show the lowest RMSE. Due to the high computational requirements of weighted thin plate smoothing, we omit post processing from [35]. Li et al. [31, 32, 30] propose a rectification to allow for standard stereo algorithms to run on spherical images. For fair comparison, we paired it with a more recent and highly optimized SGM implementation on GPU [18]. We also implement Lin et al.'s [33] method on GPU.

Table 3 shows that our method significantly outperforms both analytical and data driven methods on our rendered dataset while being orders of magnitude more computationally efficient. In addition, as the three depth from spherical image pair methods [56, 35, 32] are unable to estimate depth in the baseline's alignment, the distance estimation quality is highly degraded, hence their substantial >0.4 bad pixel ratio. We do not evaluate the panorama quality for these method as we use the rendered panoramas as input, ignoring possible stitching artifacts that may occur in real world scenarios. Figures 7, 9 and 10 show that our method is able to estimate distance accurately using our prototype, while providing a convincing panorama. Refer to the supplemental document for additional comparisons.

5. Conclusion

Our work achieves real-time omnidirectional RGB-D imaging directly from fisheye images. The method does not rely on temporal information nor on special illumination making it robust to changing scenes and suitable for both indoor and outdoor situations. Thanks to our adaptive spherical matching and fast inter-scale bilateral filtering, we demonstrate a combination of accuracy and performance suitable for interactive and robotic applications in dynamic environments.

* CrownConv inference runs with a fixed number of vertices (10242).
† no reference implementation is provided.
‡ part of the method runs on CPU.

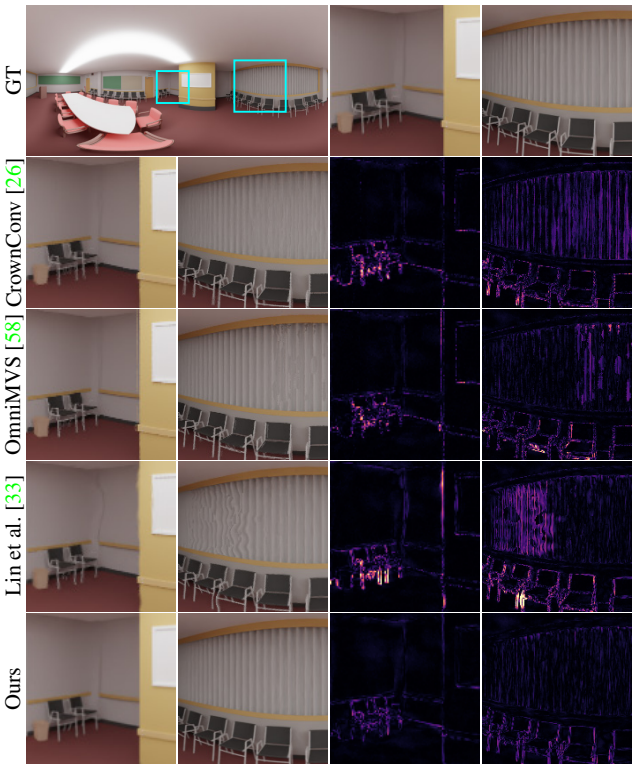


Figure 10: Image stitching results on our synthetic dataset.

	Inverse distance (% | m^{-1})				Panorama		Runtime
	>0.1	>0.4	MAE	RMSE	PSNR	SSIM	(ms)
CrownConv* [26]	57.09	2.53	0.135	0.168	36.35	0.985	$5.2{\cdot}10^{2}$
OmniMVS† [58]	39.01	5.64	0.124	0.182	37.28	0.986	$1.3{\cdot}10^{3}$
Sweepnet‡ [59]	37.13	3.00	0.101	0.133	35.46	0.981	$1.0{\cdot}10^{5}$
Lin et al.† [33]	37.25	5.13	0.181	0.181	36.22	0.980	$2.5{\cdot}10^{3}$
360SD-Net [56]	54.29	13.27	0.212	0.341	–		$6.1{\cdot}10^{2}$
Matzen et al.†‡ [35]	29.59	18.64	0.170	0.275	–		$9.3{\cdot}10^{1}$
Li + SGM [32, 18]	31.04	16.85	0.170	0.283	–		$5.8{\cdot}10^{0}$
Ours	**20.38**	**0.56**	**0.068**	**0.095**	**38.78**	**0.990**	$\mathbf{2.8{\cdot}10^{0}}$

Table 3: Comparison of our spherical RGB-D results against other methods. All methods are run on our desktop test system and output a 1024×512 px distance map. Note that since [56, 35, 32] inputs are ground truth panoramas, the image quality is not directly comparable.

Acknowledgments

Min H. Kim acknowledges Samsung Research Funding Center (SRFC-IT2001-04) for developing partial 3D imaging algorithms, in addition to partial support of Korea NRF grants (2019R1A2C3007229, 2013M3A6A6073718), MSIT/IITP of Korea (2017-0-00072), and MSRA.

References

[1] Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernandez Esteban, Sameer Agarwal, and Steven M. Seitz. Jump: Virtual reality video. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2016.

[2] Z. Arican and P. Frossard. Dense disparity estimation from omnidirectional images. In *2007 IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 399–404, 2007.

[3] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017.

[4] Jonathan T. Barron and Ben Poole. The fast bilateral solver. *ECCV*, 2016.

[5] Tobias Bertel, Mingze Yuan, Reuben Lindroos, and Christian Richardt. OmniPhotos: Casual 360° VR photography. *ACM Transactions on Graphics*, 39(6):266:1–12, Dec. 2020.

[6] Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics (TOG)*, 39(4):86–1, 2020.

[7] D. Caruso, J. Engel, and D. Cremers. Large-scale direct slam for omnidirectional cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 141–148, 2015.

[8] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[9] Jiawen Chen, Andrew Adams, Neal Wadhwa, and Sam Hasinoff. Bilateral guided upsampling. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)*, 2016.

[10] Jiawen Chen, Sylvain Paris, and Frédo Durand. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph.*, 26(3):103–es, July 2007.

[11] Zhaopeng Cui, Lionel Heng, Y. Yeo, Andreas Geiger, M. Pollefeys, and Torsten Sattler. Real-time dense mapping for self-driving vehicles using fisheye cameras. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6087–6093, 2019.

[12] F. Devernay and O. Faugeras. Straight lines have to be straight. In *Machine Vision and Applications*, 2001.

[13] W. Gao and S. Shen. Dual-fisheye omnidirectional stereo. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6715–6722, 2017.

[14] Eduardo S. L. Gastal and Manuel M. Oliveira. Domain transform for edge-aware image and video processing. *ACM TOG*, 30(4):69:1–69:12, 2011. Proceedings of SIGGRAPH 2011.

[15] Kaiming He and Jian Sun. Fast guided filter. *ArXiv*, abs/1505.00996, 2015.

[16] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 1–14, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[17] S. Hermann and T. Vaudrey. The gradient - a powerful and robust cost function for stereo matching. In *2010 25th International Conference of Image and Vision Computing New Zealand*, pages 1–8, 2010.

[18] Daniel Hernandez-Juarez, Alejandro Chacón, Antonio Espinosa, David Vázquez, Juan Carlos Moure, and Antonio M. López. Embedded real-time stereo estimation via semi-global matching on the GPU. In *International Conference on Computational Science 2016, ICCS 2016, 6-8 June 2016, San Diego, California, USA*, pages 143–153, 2016.

[19] H. Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814 vol. 2, 2005.

[20] T. Ho, I. D. Schizas, K. R. Rao, and M. Budagavi. 360-degree video stitching for dual-fisheye lens cameras based on rigid moving least squares. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 51–55, 2017.

[21] Tuan Anh Ho and Madhukar Budagavi. Dual-fisheye lens stitching for 360-degree imaging. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2172–2176, 2017.

[22] Christian Häne, Lionel Heng, Gim Lee, Alexey Sizov, and Marc Pollefeys. Real-time direct dense matching on fisheye images using plane-sweeping stereo. pages 57–64, 02 2015.

[23] Sunghoon Im, Hyowon Ha, François Rameau, Hae-Gon Jeon, Gyeongmin Choe, and In So Kweon. All-around depth from small motion with a spherical panoramic camera. In *European Conference on Computer Vision*, pages 156–172. Springer, 2016.

[24] Jose M. Alvarez Jiayu Yang, Wei Mao and Miaomiao Liu. Cost volume pyramid based depth inference for multi-view stereo. In *CVPR*, 2020.

[25] J. Kannala and S. S. Brandt. A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1335–1340, 2006.

[26] Ren Komatsu, Hiromitsu Fujii, Yusuke Tamura, Atsushi Yamashita, and Hajime Asama. 360° depth estimation from multiple fisheye images with origami crown representation of icosahedron. In *IROS*, 2020.

[27] Johannes Kopf, Michael Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, volume 26. Association for Computing Machinery, Inc., August 2007.

[28] Po Kong Lai, Shuang Xie, Jochen Lang, and Robert Laqaruère. Real-time panoramic depth maps from omni-directional stereo images for 6 DOF videos in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 405–412. IEEE, 2019.

[29] Jongchul Lee, Daeyoon Jun, Changkyoung Eem, and Hyunki Hong. Improved census transform for noise robust stereo matching. *Optical Engineering*, 55(6):1 – 10, 2016.

[30] Shigang Li. Binocular spherical stereo. *IEEE Transactions on intelligent transportation systems*, 9(4):589–600, 2008.

[31] Shigang Li and Kiyotaka Fukumori. Spherical stereo for the construction of immersive VR environment. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.*, pages 217–222. IEEE, 2005.

[32] Li, Shigang. Real-time spherical stereo. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 1046–1049, 2006.

[33] Hong-Shiang Lin, Chao-Chin Chang, Hsu-Yu Chang, Yung-Yu Chuang, Tzong-Li Lin, and Ming Ouhyoung. A low-cost portable polycamera for stereoscopic 360° imaging. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[34] Chuiwen Ma, Liang Shi, Hanlu Huang, and Mengyuan Yan. 3d reconstruction from full-view fisheye camera. *arXiv preprint arXiv:1506.06273*, 2015.

[35] Kevin Matzen, Michael F Cohen, Bryce Evans, Johannes Kopf, and Richard Szeliski. Low-cost 360 stereo photography and video capture. *ACM Transactions on Graphics (TOG)*, 36(4):148, 2017.

[36] Amrita Mazumdar, Armin Alaghi, Jonathan T. Barron, David Gallup, Luis Ceze, Mark Oskin, and Steven M. Seitz. A hardware-friendly bilateral solver for real-time virtual reality video. In *Proceedings of High Performance Graphics*, HPG '17, New York, NY, USA, 2017. Association for Computing Machinery.

[37] Morgan McGuire. Computer graphics archive, July 2017. `https://casual-effects.com/data`.

[38] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015.

[39] Branislav Micusík and Tomás Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 485–490, 2003.

[40] Simon Niklaus and Feng Liu. Softmax splatting for video frame interpolation. In *CVPR*, 2020.

[41] Manuel Oliveira, Brian Bowen, Richard McKenna, and Yu-Sung Chang. Fast digital image inpainting. *VIIP*, pages 261–266, 01 2001.

[42] Sarthak Pathak, Alessandro Moro, Atsushi Yamashita, and Hajime Asama. Dense 3D reconstruction from two spherical images via optical flow-based equirectangular epipolar rectification. In *2016 IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 140–145. IEEE, 2016.

[43] Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*. ACM, 2017.

[44] Albert Pozo, Michael Toksvig, Terry Schrager, Joyce Hsu, Uday Mathur, Alexander Sorkine-Hornung, Rick Szeliski, and Brian Cabral. An integrated 6DoF video camera and system design. *ACM Transactions on Graphics*, 38:1–16, 11 2019.

[45] Menandro Roxas and Takeshi Oishi. Real-time variational fisheye stereo without rectification and undistortion. *arXiv preprint arXiv:1909.07545*, 2019.

[46] M. Roxas and T. Oishi. Variational fisheye stereo. *IEEE Robotics and Automation Letters*, 5(2):1303–1310, 2020.

[47] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, pages 131–140, 2001.

[48] M. Schönbein and A. Geiger. Omnidirectional 3d reconstruction in augmented manhattan worlds. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 716–723, 2014.

[49] Christophe Seux. Blender demo files, November 2020. `blender.org/download/demo-files/`.

[50] Tomáš Svoboda, Tomáš Pajdla, and Václav Hlaváč. Epipolar geometry for panoramic cameras. In Hans Burkhardt and Bernd Neumann, editors, *ECCV*, 1998.

[51] Xiao Tang, Xiaowei Hu, Chi-Wing Fu, and Daniel Cohen-Or. GrabAR: Occlusion-aware grabbing virtual objects in AR. *UIST*, 2020.

[52] Akihiko Torii, Atsushi Imiya, and Naoya Ohnishi. Two- and three- view geometry for spherical cameras. In *workshop on omnidirectional vision, camera networks and non-classical cameras*, 2005.

[53] V. Usenko, N. Demmel, and D. Cremers. The double sphere camera model. In *Proc. of the Int. Conference on 3D Vision (3DV)*, September 2018.

[54] Fu-En Wang, Hou-Ning Hu, Hsien-Tzu Cheng, Juan-Ting Lin, Shang-Ta Yang, Meng-Li Shih, Hung-Kuo Chu, and Min Sun. Self-supervised learning of depth and camera motion from 360° videos. *ACCV*, 2018.

[55] Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. BiFuse: Monocular 360 depth estimation via bi-projection fusion. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[56] Ning-Hsu Wang, Bolivar Solarte, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. 360SD-Net: 360° stereo depth estimation with learnable cost volume. In *ICRA*, 2020.

[57] Yan Wang, Zihang Lai, Gao Huang, Brian H. Wang, Laurens Van Der Maaten, Mark Campbell, and Kilian Q Weinberger. Anytime stereo image depth estimation on mobile devices. *International Conference on Robotics and Automation (ICRA)*, 2019.

[58] Changhee Won, Jongbin Ryu, and Jongwoo Lim. OmniMVS: End-to-end learning for omnidirectional stereo matching. In *ICCV*, 2019.

[59] Changhee Won, Jongbin Ryu, and Jongwoo Lim. SweepNet: Wide-baseline omnidirectional depth estimation. In *ICRA*, 2019.

[60] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. TossingBot: Learning to throw arbitrary objects with residual physics. 2019.

[61] K. Zhang, Y. Fang, D. Min, L. Sun, S. Yang, and S. Yan. Cross-scale cost aggregation for stereo matching. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(5):965–976, 2017.

[62] Nikolaos Zioulis, Antonis Karakottas, Dimitrios Zarpalas, and Petros Daras. Omnidepth: Dense depth estimation for indoors spherical panoramas. In *ECCV*, pages 453–471, 2018.