Fast Omnidirectional Depth Densification

Hyeonjoong Jang, Daniel S. Jeon, Hyunho Ha, and Min H. Kim

KAIST School of Computing

Abstract. Omnidirectional cameras are commonly equipped with fisheye lenses to capture 360-degree visual information, and severe spherical projective distortion occurs when a 360-degree image is stored as a two-dimensional image array. As a consequence, traditional depth estimation methods are not directly applicable to omnidirectional cameras. Dense depth estimation for omnidirectional imaging has been achieved by applying several offline processes, such as patch-matching, optical flow, and convolutional propagation filtering, resulting in additional heavy computation. No dense depth estimation for real-time applications is available yet. In response, we propose an efficient depth densification method designed for omnidirectional imaging to achieve 360-degree dense depth video with an omnidirectional camera. First, we compute the sparse depth estimates using a conventional simultaneous localization and mapping (SLAM) method, and then use these estimates as input to a depth densification method. We propose a novel densification method using the spherical pull-push method by devising a joint spherical pyramid for color and depth, based on multi-level icosahedron subdivision surfaces. This allows us to propagate the sparse depth continuously over 360-degree angles efficiently in an edge-aware manner. The results demonstrate that our real-time densification method is comparable to state-of-the-art offline methods in terms of per-pixel depth accuracy. Combining our depth densification with a conventional SLAM allows us to capture real-time 360-degree RGB-D video with a single omnidirectional camera.

Keywords: omnidirectional stereo · 3D imaging · depth densification.

1 Introduction

Omnidirectional cameras have been popularly used for capturing 360-degree images and are essential for virtual reality (VR)/augmented reality (AR) applications to mix rendered virtual objects in real scenes. Omnidirectional cameras are equipped with fisheye-lenses to capture a very wide field of view (FOV) with even more than 180 degrees for each lens, yielding 360-degree images with a single camera. Different from ordinary 2D images, 360-degree images have special properties that give rise to new **challenges** in terms of image processing. First, in order to store 360-degree image data in the conventional 2D image array, a geographic projection mapping between spherical and image coordinates is required additionally: e.g., equirectangular projection, latitude-longitude projection, cube-map projection, concentric projection, etc. [22]. However, when

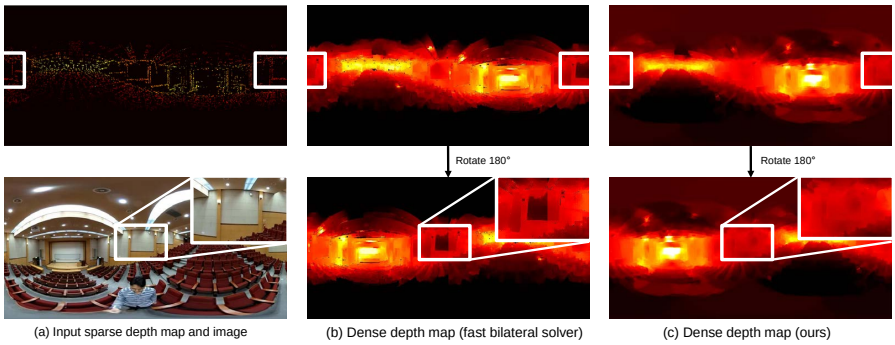

Fig. 1. (a) shows an example of a sparse depth map as input obtained from a visual SLAM method [5] and an equirectangular color image captured by an omnidirectional camera. (b) presents a densification result in two different orientations using the fast bilateral solver [1] on the equirectangular depth map. When the 360-degree image is rotated, the simple densification on the equirectangular image suffers from the seam artifacts. (c) shows our densification result. Our densification does not suffer from seam artifacts when the depth map is rotated in a different orientation. The closeup of the white/beige-colored wall compares the artifacts for the two methods.

360-degree image data are geographically projected and stored as a 2D image array, severe geometrical *warp* inevitably occurs in the stored 360-degree image; for example, a straight line of the object may appear warped like an arc in the equirectangular image (see the white arc in the middle in Figure 1 as an example), where certain parts are distorted, enlarged, or shrunk unevenly and nonlinearly depending on their geometric positions. In particular, spherical distortion in geographic projection mapping is a severe problem when applying existing image processing operators for 360-degree images as they have been developed for ordinary 2D images. These challenges have been resolved by additionally applying several offline processes, such as patch-matching, optical flow, and convolutional propagation filtering, resulting in additional heavy computation. To the best of our knowledge, no fast depth densification method for 360-degree images is available yet.

Second, the ordinary 2D image data have four boundaries: the leftmost, rightmost, topmost, and bottommost ends of the image. In contrast, the 360-degree image data have *no ends*; i.e., parts of the image are *circularly connected without ends*. For example, when a patch-wise operator is applied for depth densification on an unfolded 360-degree image (such as an equirectangular image, shown in Figure 1), the leftmost and rightmost parts are different after applying the densification operator. The difference becomes clear when the processed depth map is wrapped in the 360-degree image data. The traditional algorithms are not free from the inconsistency artifacts because they cannot satisfy the circular constraints when the results are wrapped in 360-degree images.

Figure 1 compares depth densification results from the sparse depth input (a) using a state-of-the-art method, the fast bilateral solver [1] (b) and ours (c). The top rows of (b) and (c) show densification results by two different

methods, and the bottom rows below present 180-degree rotated images of the same scene. The top depth map in (b) is densified from the left sparse, an equirectangular depth map by the densification method, and the map below in (b) shows the 180-degree rotation of the same propagated depth map. Note that the leftmost and rightmost regions indicate the same area, the white/beige-colored wall, in (b) and (c). When the resulting 360-degree images are rotated, the naïve densification method presents seam artifacts on the stitched area due to inconsistent computation over edges.

In this work, we propose an efficient dense densification method specially designed for omnidirectional imaging such that it allows the 360-degree dense depth video to be captured with a single omnidirectional camera. First, we obtain sparse depth estimates per each subframe from the input 360-degree video using a real-time simultaneous localization and mapping (SLAM) method [5] as input for our densification method. As our key contribution, we introduce a novel spherical pull-push method by means of a *joint spherical pyramid* for color and depth information, based on multi-level icosahedron subdivision surfaces. Our method accounts for not only the aforementioned characteristics of 360-degree images (i.e., distortion and circularity), but also is computationally efficient. It takes just an additional 15 milliseconds for propagating the sparse depth estimates to every pixel with awareness of edge structures using a conventional GPU. Combining our depth densification with the conventional SLAM allows us to capture 360-degree dense depth video in real-time with a single omnidirectional camera.

2 Related Work

Multi-camera methods In order to cover the entire field of view in 360-degree angles, several multi-camera omnidirectional methods have been proposed. They are equipped with an array of multiple cameras with ordinary lenses on a circle structure [2, 24]. They then apply the traditional stereo matching algorithm, assuming a cylindrical projection model with an ordinary field of view. They can search stereo correspondence based on epipolar geometry through rectification. However, these methods can capture depth along the azimuthal angles only. Their results cannot obtain complete 360-degree depth information, due to the lack of cameras and image formation at the top and bottom directions.

The form factor of multi-camera systems is significantly increased by having many cameras and consequently they are not portable. With efforts focused on reducing the form factor, multiple fisheye lenses have been installed in omnidirectional camera systems [16–18]. The FOV of fisheye lenses is significantly larger at about $180° - 190°$ than those of ordinary lenses. Thus the pinhole-based perspective projection model is invalid with fisheye lenses. They, therefore, employ the spherical rectification method, which is devised on the inverse-equirectangular projection model, a.k.a. the latitude-longitude (LL) projection model. On the rectified images in the spherical domain, they use the conventional stereo algorithm, scanning stereo correspondence along the horizontal line on a pair of LL

projection images. They produce a complete 360-degree depth map, but the computation is time consuming. For instance, Lin et al. [18] took about 12 minutes for rectification and depth estimation for processing one 360-degree image.

Single-camera methods Recently, extensive efforts have been made to enable *monoscopic* 360-degree depth imaging. Caruso et al. [3] introduced a visual SLAM method for a hemispherical fisheye-lens camera that covers 185° angles. They estimate sparse depth points and propagate them partially while tracking the camera motion. Their method is devised for navigation using a fisheye-lens camera and therefore they only provide partially depth information, which is not applicable for 360-degree image-based rendering. Im et al. [12] and Huang et al. [11] make use of a single 360-degree RGB camera (equipped with two front/back fisheye lenses). They both take video frames as input and apply the structure-from-motion algorithm to estimate the camera pose parameters. Im et al. densify the sparse depth using the sphere-sweeping method based on the stereo algorithm. Huang et al. propagate the sparse depth using Delaunay Triangulation-based densification [21]. However, similar to the previous spherical stereo methods, these monoscopic methods also require more than 10 minutes to densify the sparse depth for one omnidirectional depth map. The proposed method is 40,000 times faster than the state-of-the-art methods of omnidirectional depth densification, reducing the computational time from 10 minutes to 15 milliseconds.

Depth densification for traditional 2D images Depth densification is a long-lasting problem and has been researched for many years in computer vision. For brevity, we simplify the overview of depth densification methods for traditional 2D images. Sparse depth information as points or edges has been propagated to every pixel in a dense depth map using the colorization solver [15], the guided image filter [9], or the fast bilateral solver [1]. Different from the simple interpolation of sparse point, the main objective of these methods is to densify the sparse information while preserving sharp edges of objects in the scene. Densification is time consuming in nature [15, 8]. To address the speed, a fast bilateral solver [1] and a fast depth densification [10] were also proposed for traditional 2D images, in addition to an optimization approach [8] and neural network-based inference [19, 4, 13]. However, all these methods are suitable for traditional 2D images, and are not directly applicable for the projected views of 360-degree image data, resulting in inconsistency artifacts (as shown in Figure 1). In contrast, we propose a novel depth densification method using a joint spherical pyramid on a multiscale architecture with icosahedron subdivision, which satisfies the characteristics of 360-degree images.

3 Omnidirectional Depth Densification

3.1 360-degree Sparse Depth Estimation

We capture 360-degree video using a conventional omnidirectional camera and then obtain sparse depth information and extrinsic camera parameters for each

frame using a visual SLAM method [5]. We use the sparse depth information of the 360-degree depth map as input for our densification method.

360-degree to 2D images The visual SLAM method that we used is a fast open source method, but is designed for traditional 2D images [5], rather than 360-degree images. The camera stores the input video in the equirectangular format. When the SLAM method runs on the original format of 360-degree video, the number of detected features decreases significantly due to geographic distortion by spherical projection. To address this challenge, we project the input 360-degree video frames to the cube-map representation model, as shown in Figure 2. We first divide each input 360-degree image into six-face images that look outward in six orthogonal directions to each other in the spherical domain.

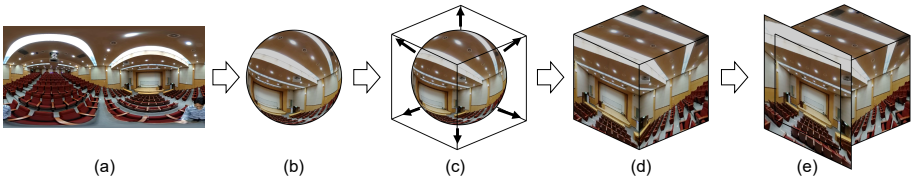

Fig. 2. (a) An equirectangular image that represents a 360-degree image. (b) A sphere mapping of the equirectangular image. (c) Projection from the sphere to a cube. (d) A cube map of the 360-degree image. (e) The extension of each face with FOV margins for robust depth estimation near borders.

Depth estimation It is worth noting that when we subdivided and projected to six faces (shown in (d)) and ran the SLAM method for six cameras toward each face, we found that SLAM features cannot be captured well near square boarders of each face. We extended the camera FOV from 90 degrees to 102 degrees for each face direction. Once we could obtain features well within the square face, we cut out the extra margins when they were combined every detected feature to a 360-degree depth map via the spherical representation. However, since we estimate sparse depth on each face separately, the depth scales from each face camera may be different from each other. Thus we normalize depth values in each face camera using the magnitude of the six estimated camera translation vectors and depth values overlapped in the extra margins. This allows us to capture sparse features well with the omnidirectional camera input and traditional 2D SLAM method. This preprocess can be substituted with any available 360-degree SLAM method [3].

3.2 Omnidirectional Depth Densification

Now we describe a fast depth densification method that densifies the sparse depth to obtain a complete 360-degree depth map using our spherical pull-push algorithm.

Joint Spherical Pyramid Since we estimate the sparse depth for the six faces in the cube map, we then project them to the unit sphere domain, as shown in the top-left sphere in Figure 3(a). The spherical image pyramid was initially proposed to achieve scale-invariance for effectively detecting visual features in omnidirectional SLAM methods [7, 23]. In contrast, we devise a novel spherical architecture for fast depth densification. We propose a *joint spherical pyramid* that includes multiscale color and depth information in the spherical domain by subdividing icosahedron in multiple resolutions, where each level of pyramid corresponds to each level of subdivision.

Figure 3(a) shows an example of our joint spherical pyramid. For example, level 0 represents a 20-face icosahedron and level 1 represents an 80-face structure[1]. The number of triangles increases by four times when the level increases; i.e., four triangles in level n+1 corresponds to one triangle in level n (see Figure 3b). Suppose we subdivide the 20-face icosahedron up to the 8th level. The number of polygons increases to $20\times4^8 = 1{,}310{,}720$ faces, the number of which is similar to the number of pixels in a cube map with 6 faces of each face resolution of 480×480 (1,382,400 pixels in total). We set the finest level of subdivision to level 8 for our experiment.

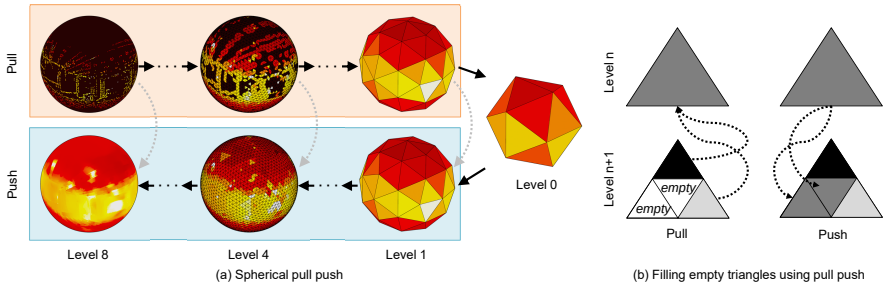

Fig. 3. (a) Spherical pull-push algorithm using multiple subdivision levels of icosahedron. (b) Four triangles in the current level corresponds to one triangle in the upper level. We compute the mean of existing values in the pull phase, and fill empty triangles with the mean value in the push phase.

Lookup tables Our method allows for precomputing the geographic relationship between pixels and polygons of the spherical pyramid. We make use of two precomputed lookup tables: (a) each pixel in the cube map to each polygon in each level of the spherical pyramid, and (b) nearest neighbors for each polygon at each level. Unlike in 2D images, we need to define the neighbor set N with nearest neighbors as we use the spherical structure with triangle polygons. Since we know every 3D position of triangles on the unit sphere, we can find neighbors by calculating the Euclidean distance. The number of polygons is very large in

[1] Note that contrary to the labeling convention of the image pyramid, we label each level from the coarse to fine level in ascending order, following the subdivision labeling convention.

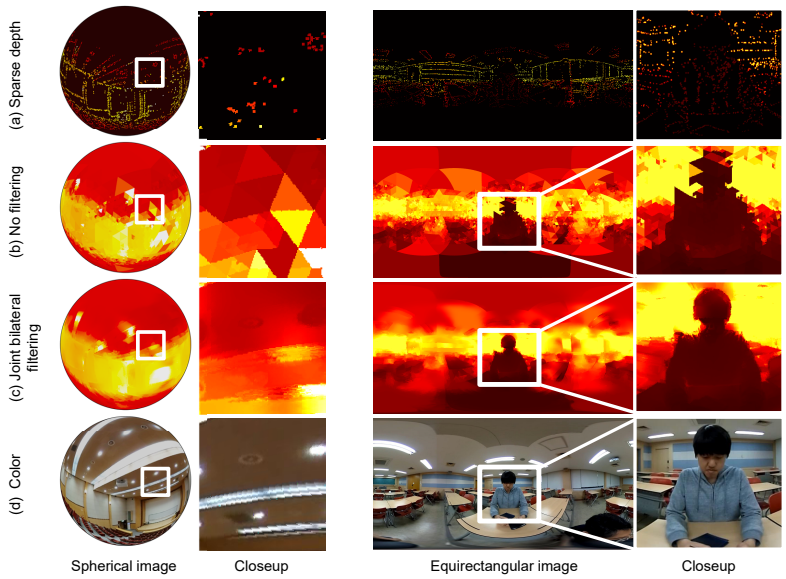

Fig. 4. (a) shows the sparse depth input. (b) presents the initial depth map filled by the pull-push process. (c) In the push phase, we refine the initial depth map using the joint bilateral filter at each level to remove triangle artifacts, starting from level 5 to 8. It refines and preserves edges in the final depth map. (d) shows the guide color images used for filtering.

our case. Thus, we precompute the neighbors using an approximating nearest neighbor algorithm [20] and save them as a lookup table for each level in advance.

Spherical Densification Our densification procedure consists of two main parts: First, we fill empty depth pixels coarsely using the pull-push algorithm [6] in our joint spherical pyramid. Second, we refine the coarsely filled depth values to have fine edge details with joint bilateral filtering [14] at each fine level of the spherical pyramid.

Our pull-push algorithm has two phases, the pull phase and the push phase. In the pull phase, we compute the average of existing depth values for every four triangles at the level n+1 and assign the average to the polygon at the level n. See Figure 3(b). In case none of the four triangles at the level n+1 have depth values, we treat the triangle as empty at the n-level. We found that setting level 0 with the 20-face icosahedron can fill every polygon at the end of the pull phase with natural scenes. In the push phase, we fill every empty triangle in the n+1 level with the average depth values of the coarse triangles in the n level. Note that we do not change existing values of the triangle in the push phase. This allows us to fill empty regions while preserving low-frequency edge structures.

Edge-aware filtering In order to preserve high-frequency details, we refine depth values with joint bilateral filtering [14] on the spherical space at each level before

pushing the current depth values to the next finer level. For each triangle p and its neighbor triangle q in the neighbor field N, the refined depth $\tilde{D}_p$ is calculated by the Gaussian weighted sum of neighbor depths D_q. We employ two different weights: spatial distance in the spherical space and the color difference as follows:

$$\widetilde{D}_p = \frac{1}{W_p} \sum_{q \in N} \exp\left(-\frac{\|X_p - X_q\|_2^2}{2\sigma_s} - \frac{\|C_p - C_q\|_2^2}{2\sigma_c} \right), \tag{1}$$

where X_p and C_p are a 3D coordinate vector and an RGB color vector of triangle p, respectively. In addition, σ_s and σ_c are the parameters for spatial distance and color difference, respectively, and W_p is the normalization constant, the sum of the products of the Gaussian weights. We found that filtering the coarse depth information at level 0 to 4 does not provide positive effects because each triangle covers an excessively large area. Thus, we apply the filter from level 5 to level 8 (finest) only. Lastly, we reproject the dense depth values in the cube map into the final output format, the equirectangular image. In addition, we apply the temporal median filter among three neighboring frames (previous/current/next) to reduce noise in the densified depth video. See Figure 4 for an example.

4 Results

We tested our algorithm with 360-degree videos (1920×960 pixels) captured by a Ricoh Theta camera. Our algorithm is implemented using C++ with CUDA. We used a machine with an Intel i7-6700 4.00 GHz processor with 32 GB RAM and NVIDIA GeForce GTX 1080 Ti. We configured a total of eight levels of subdivisions of an icosahedron to compose the joint spherical pyramid. In the push phase, the joint bilateral filter is applied from level 5 to level 8 only. The color weight parameter σ_c is set to 4.02 and the spatial parameter σ_s is set to 40.2. N is set to 500 to ensure the filter preserves edges effectively. Our algorithm takes less than 15 milliseconds with our CUDA implementation to densify a sparse depth map obtained from the visual SLAM method [5].

For a quantitative evaluation of our algorithm, we rendered two virtual scenes and captured the ground-truth (GT) depth in the equirectangular image format. We also obtained sparse depth maps from the virtual scenes as input for densification algorithms. We compared the accuracy of our method compared (15 milliseconds per frame on GPU) with other densification methods: fast bilateral solver (0.49 second per frame on CPU) [1], colorization solver (96.83 seconds per frame on CPU) [15], and guided image filter (0.36 second per frame on CPU) [9]. We used the authors' original publicly available implementations, which are written in Python. Therefore, performance is not directly comparable. As shown in Table 1, our method provides accurate dense depth maps compared to other methods. We measured the mean squared error (MSE) values for each result with GT. The colorization solver and ours show the highest accuracy among four methods. However, as shown in Figure 5, the colorization solver presents severe color dependency when densifying depth and also cannot preserve high-frequency details in dense depth maps, compared to our method.

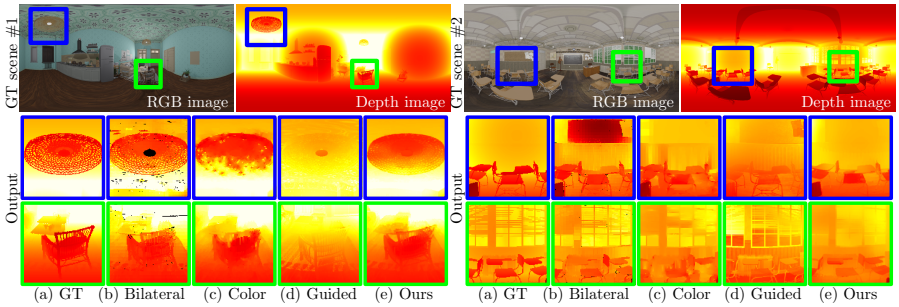

Fig. 5. Closeup results compared to the ground-truth image. (a) Ground-truth depth image. (b) Fast bilateral solver. (c) Colorization solver. (d) Guided image filter. (e) Ours.

Method	Fast bilateral solver	Colorization solver	Guided image filter	Ours
GT scene #1	0.5758	0.0448	0.2040	0.0336
GT scene #2	0.3764	0.0518	0.2463	0.1659

Table 1. The MSE errors between each result (Figure 5) and the ground-truth depth map with the four different methods. The colorization solver and our method provide the highest accuracy. However, the colorization solver cannot preserve edge structures well, while our method preserves high-frequency edge structures clearly with overall high accuracy.

Figure 6 presents dense depth map results from real monoscopic, omnidirectional camera input. Our method outperforms other densification methods in two aspects: First, as shown in the first row of closeups, our method can propagate sparse depth more independently of color properties than the compared methods. Second, as shown in the second row of results, our method is free from the seam artifacts at the stitched border of the 360-degree image. This is because our method takes the spherical characteristics into account. Note that the flickering artifacts in videos are originated from the inaccurate depth estimates of the used SLAM method [5].

Finally, we compared our results with the monocular, omnidirectional dense depth imaging method proposed by Im et al. [12]. Our method is based on the SLAM input and allows for dynamic motion, yielding real-time dense depth output. However, 30 frames with *small* motion are required to estimate one depth frame. Therefore, their method is not applicable to large motion. Moreover, their computation is time consuming. It took 5 minutes per frame, thus not allowing for real-time applications. Figure 7 qualitatively compares dense depth maps captured by two methods.

5 Conclusion

We have presented a novel depth densification algorithm using a joint spherical pyramid that considers color and depth simultaneously. The joint spherical

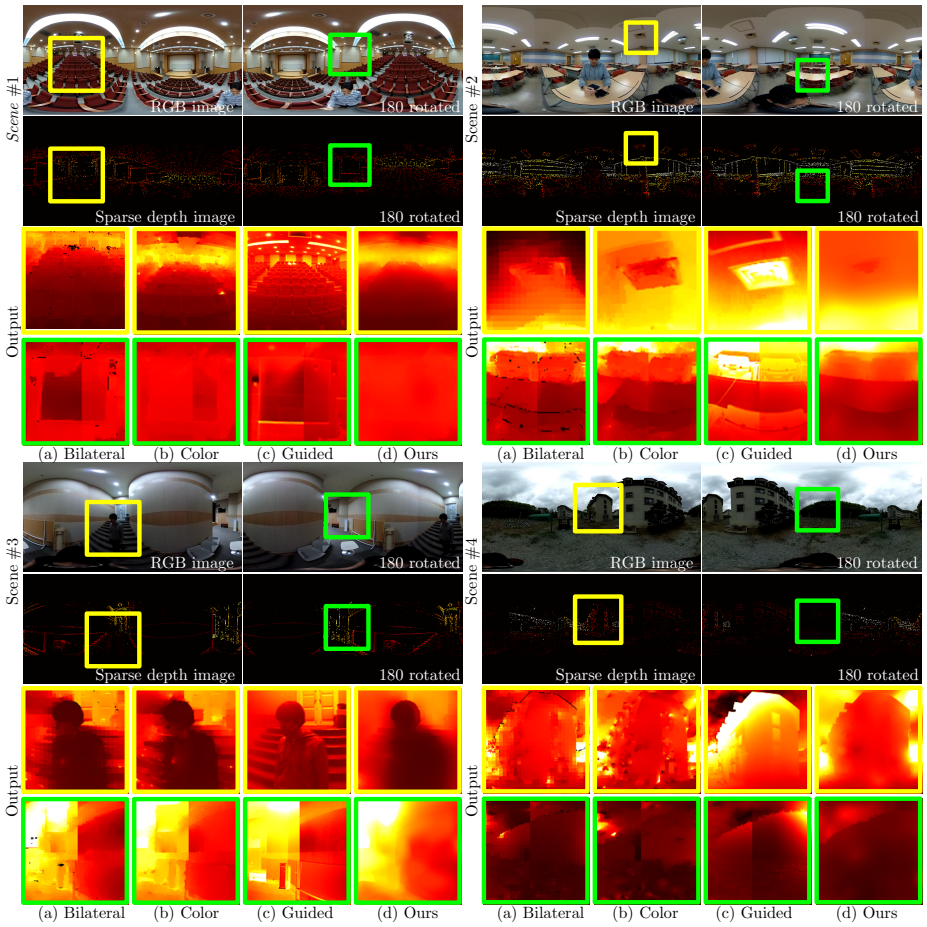

Fig. 6. Comparison results with monoscopic real camera input. (a) Fast bilateral solver. (b) Colorization solver. (c) Guided image filter. (d) Ours.

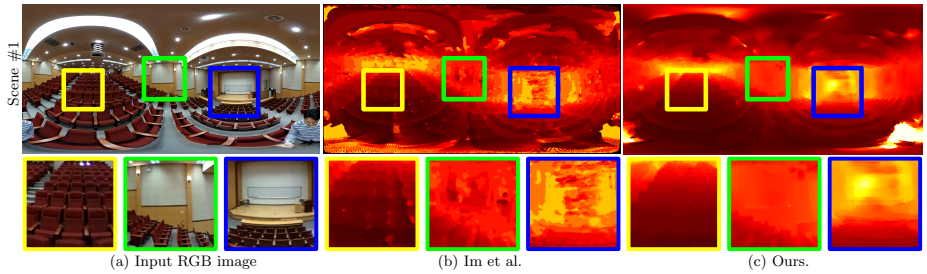

Fig. 7. Comparison with other monocular, omnidirectional dense depth imaging. (a) Input image. (b) Im et al. [12]. (c) Ours.

pyramid is made of a total of eight levels of subdivision of a 20-face icosahedron. Our algorithm consists of two phases; in the pull phase, we average up the sparse depth values and store them in the spherical pyramid from the fine to the coarse level. In the push phase, we filled empty polygons from the averaged depth values of the upper level of the pyramid. In order to preserve high-frequency details, we filter each level with the joint bilateral filter with the input color image. Our method accounts for the characteristics of the 360-degree images and as such it shows no seam effects of propagation. Moreover, our method is computationally efficient, taking less than 15 milliseconds, because it is parallelizable and implemented with CUDA and make use of precomputed lookup tables. It is adoptable for other real-time VR/AR applications that require dense depth maps for handling occlusion in rendering.

Acknowledgements

Min H. Kim acknowledges Korea NRF grants (2019R1A2C3007229, 2013M3A6A-6073718) and additional support by Cross-Ministry Giga KOREA Project (GK17-P0200), Samsung Electronics (SRFC-IT1402-02), ETRI(19ZR1400), and an ICT R&D program of MSIT/IITP of Korea (2016-0-00018).

References

1. Barron, J.T., Poole, B.: The fast bilateral solver. In: European Conference on Computer Vision. pp. 617–632. Springer (2016)
2. Bunschoten, R., Krose, B.: Robust scene reconstruction from an omnidirectional vision system. IEEE Transactions on Robotics and Automation **19**(2), 351–357 (2003)
3. Caruso, D., Engel, J., Cremers, D.: Large-scale direct slam for omnidirectional cameras. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 141–148. IEEE (2015)
4. Chen, Z., Badrinarayanan, V., Drozdov, G., Rabinovich, A.: Estimating depth from rgb and sparse sensing. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 167–182 (2018)
5. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE transactions on pattern analysis and machine intelligence **40**(3), 611–625 (2018)
6. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Siggraph. vol. 96, pp. 43–54 (1996)
7. Guan, H., Smith, W.A.: Brisks: Binary features for spherical images on a geodesic grid. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4516–4524 (2017)
8. Hawe, S., Kleinsteuber, M., Diepold, K.: Dense disparity maps from sparse disparity measurements. In: 2011 International Conference on Computer Vision. pp. 2126–2133. IEEE (2011)
9. He, K., Sun, J., Tang, X.: Guided image filtering. IEEE transactions on pattern analysis and machine intelligence **35**(6), 1397–1409 (2013)
10. Holynski, A., Kopf, J.: Fast depth densification for occlusion-aware augmented reality. In: SIGGRAPH Asia 2018 Technical Papers. p. 194. ACM (2018)

11. Huang, J., Chen, Z., Ceylan, D., Jin, H.: 6-dof vr videos with a single 360-camera. In: 2017 IEEE Virtual Reality (VR). pp. 37–44. IEEE (2017)
12. Im, S., Ha, H., Rameau, F., Jeon, H.G., Choe, G., Kweon, I.S.: All-around depth from small motion with a spherical panoramic camera. In: European Conference on Computer Vision. pp. 156–172. Springer (2016)
13. Jaritz, M., De Charette, R., Wirbel, E., Perrotton, X., Nashashibi, F.: Sparse and dense data with cnns: Depth completion and semantic segmentation. In: 2018 International Conference on 3D Vision (3DV). pp. 52–60. IEEE (2018)
14. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. In: ACM Transactions on Graphics (ToG). vol. 26, p. 96. ACM (2007)
15. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: ACM transactions on graphics (tog). vol. 23, pp. 689–694. ACM (2004)
16. Li, S.: Binocular spherical stereo. IEEE Transactions on intelligent transportation systems **9**(4), 589–600 (2008)
17. Li, S., Fukumori, K.: Spherical stereo for the construction of immersive vr environment. In: IEEE Proceedings. VR 2005. Virtual Reality, 2005. pp. 217–222. IEEE (2005)
18. Lin, H.S., Chang, C.C., Chang, H.Y., Chuang, Y.Y., Lin, T.L., Ouhyoung, M.: A low-cost portable polycamera for stereoscopic 360° imaging. IEEE Transactions on Circuits and Systems for Video Technology (2018)
19. Mal, F., Karaman, S.: Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1–8. IEEE (2018)
20. Muja, M., Lowe, D.G.: Scalable nearest neighbor algorithms for high dimensional data. Pattern Analysis and Machine Intelligence, IEEE Transactions on **36** (2014)
21. Shen, S.: Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. IEEE transactions on image processing **22**(5), 1901–1914 (2013)
22. Shirley, P., Chiu, K.: A low distortion map between disk and square. Journal of graphics tools **2**(3), 45–52 (1997)
23. Zhao, Q., Feng, W., Wan, L., Zhang, J.: Sphorb: A fast and robust binary feature on the sphere. International journal of computer vision **113**(2), 143–159 (2015)
24. Zhu, Z.: Omnidirectional stereo vision. In: Proceedings of the Workshop on Omnidirectional Vision, Budapest, Hungary (2001)