An Efficient Dense Depth Map Estimation Algorithm Using Direct Stereo Matching for Ultra-Wide-Angle Images

Xiuxiu Gui and Xinyu Zhang(✉)

Shanghai Key Laboratory of Trustworthy Computing, Engineering Research Center of Software/Hardware Co-design Tech and Application, East China Normal University, Shanghai, China

xyzhang@sei.ecnu.edu.cn

Abstract. We present an efficient dense depth map estimation algorithm using patch-based direct stereo matching for ultra-wide-angle images. Our algorithm takes account of the fact that the neighboring pixels inside a local patch are likely to lie on the same plane. Our algorithm propagates the "good" initial guesses to the neighboring pixels by spatial propagation, followed by a random refinement process. Therefore, this allows finding precise depth value for each point in an infinite space using a random search strategy. Our algorithm can be used to perform 3D reconstruction using the dense depth maps directly generated from ultra-wide-angle images, especially from stereo camera pairs.

Keywords: Ultra-wide-angle camera · Depth map · Patch-based stereo matching

1 Introduction

To understand and reconstruct surrounding environments, cameras play a crucial role in many applications due to the rich and comprehensive information in images [1–3]. Among various cameras, ultra-wide-angle lenses capture the large views of a surrounding environment and expect to benefit many applications like augmented reality and robotics [4,5]. For example, in virtual/augmented reality, panoramic cameras often use a combination of multiple ultra-wide-angle lenses to capture immersive environments [6]. Mounting ultra-wide-angle cameras on the vehicle also becomes a standard option for environmental perception in autonomous driving [7] and mobile robots [8]. Depth cues of the capturing scene can benefit many computer vision tasks, such as post-processing approaches of depth-of-field rendering [2,9] and image deblurring [10].

Unlike the image features generated using sparse depth cues [11,12], dense depth map estimation aims at leveraging the large field of view to perform 3D reconstruction for the entire surrounding environment. Estimating depth from a monocular ultra-wide-angle camera is considered as a challenging and ill-posed problem. Therefore, in virtual/augmented reality, multiple ultra-wide-angle lenses are often used to perform stereo matching to generate immersive

N. Magnenat-Thalmann et al. (Eds.): CGI 2022, LNCS 13443, pp. 117–128, 2022.
https://doi.org/10.1007/978-3-031-23473-6_10

depth maps. There still exhibit a few challenges for stereo matching using ultra-wide-angle lenses. First, ultra-wide-angle cameras have severe geometric distortions. Many standard stereo matching algorithms like image rectification may undergo perspective projection [13], spherical model projection [14] or equirectangular projection [15]. These projections exhibit severe distortions in the resulting depth maps and therefore this may significantly reduce the view coverage provided by the ultra-wide-angle cameras. Second, estimating correct depth values is non-trivial especially for ultra-wide-angle cameras. For example, plane sweeping [16] searches pre-defined plane hypotheses for depth values of each pixel and often leads to inaccurate depth values for pixels located at the planes that are not included in the pre-defined plane set. Note that the pre-defined possible planes are limited, though the number of candidate planes is infinite. Third, many studies can only handle ultra-wide-angle stereo pairs undergoing small displacements [17], but they may not suit for rapid movement.

Main Result: In this paper, we present a new efficient patch-based stereo matching algorithm for the ultra-wide-angle lens. We introduce the PatchMatch randomized searching method [18], which is proved to be efficient on searching the matching patch in image domain [2,19]. Patch matching stereo allows finding accurate depth values for each point in an infinite space with a random search strategy. Intuitively, patch match stereo efficiently traverses the infinite candidate planes and performs a one-shot optimization in which the planes and pixel assignments to corresponding planes are determined at the same time. This avoids the problem of missing correct planes. Our algorithm has the following novel aspects.

- The algorithm can be directly applied to ultra-wide-angle image pairs without rectification, thus providing dense depth maps for the entire environment captured by cameras.
- Using patch-based stereo matching, our algorithm can locate correct planes for each pixel with a random search strategy, which avoids missing the accurate depth values.
- Our algorithm deals with stereo pairs with an arbitrary baseline. This allows handle the rapid camera movement and perform reconstruction from stereo pairs with large displacements.
- Our algorithm is applicable for any camera projection model, as long as it has a closed-form inverse.

2 Patch Matching Stereo

In this section, we first introduce the double sphere camera projection model for ultra-wide-angle lenses. Then we demonstrate adopting the double sphere camera model in patch-based stereo matching.

2.1 Camera Projection Model

In this paper, we demonstrate the proposed algorithm with the double sphere camera model [20] since it has a closed-form inversion (unprojection) and can

avoid high computational costs. Note that our algorithm works for any ultra-wide-angle lens model that has a closed-form inversion. As shown in Sect. 3, we demonstrate some results under the unified camera model.

In the double sphere model, a point in the world space are projected onto two unit spheres consecutively. We assume that the distance between the centers of the two unit spheres is ξ. Then the point is projected onto the image plane shifted by $\frac{\alpha}{1-\alpha}$ under perspective projection. Let the camera intrinsic parameters be $\mathbf{K} = [f_x, f_y, c_x, c_y, \xi, \alpha]^T$, where $[f_x, f_y]$ are focal lengths and $[c_x, c_y]$ is the principal point given in a perspective projection. They can be typically estimated using camera calibration.

For a 3D point $\mathbf{x}$ and its corresponding image pixel $\mathbf{u}$, the function $\mathbf{u} = \pi(\mathbf{x}, \mathbf{K})$ represents the map from a 3D point onto the corresponding image plane. The unprojection function $\mathbf{x} = \pi^{-1}(\mathbf{u}, \mathbf{K})$ maps an image pixel to its corresponding viewing ray. Here, we omit the details of the double sphere camera model. We refer the readers to [20] for more explanation.

2.2 Patch Matching Stereo for Ultra-Wide-Angle Images

Patch matching stereo has been proven efficient and accurate for 3D reconstruction under perspective projection [21,22]. Here, we propose adopting an ultra-wide-angle lens projection model on patch-based stereo matching.

The proposed algorithm aims at estimating depth value based on the ultra-wide-angle stereo pair. The patch matching stereo algorithm takes input image pairs $\mathbf{I} = \{I_i, I_j\}$, together with the associating camera parameters $\{\mathbf{K}_i, \mathbf{R}_i, \mathbf{C}_i\}$ and $\{\mathbf{K}_j, \mathbf{R}_j, \mathbf{C}_j\}$, where $\mathbf{K}$ stands for camera intrinsics and $[\mathbf{R}, \mathbf{C}]$ are camera pose parameters. For a given pixel, its support plane is denoted as $[\mathbf{x}_i, \mathbf{n}_i]$, where $\mathbf{x}_i$ is a 3D point on the plane and $\mathbf{n}_i$ is the plane normal.

To generate the depth map for image I_i, we first perform a random initialization by assigning a random plane to each pixel in I_i. From the probability point of view, there must be some pixels assigned a plane very close to the correct one. Later, these good guesses will be able to propagate to the neighboring pixels by spatial propagation. The 3D point $\mathbf{x}_i$ on the random plane is defined by the scale factor of the viewing ray and can be obtained from inverse projection function given in Eq. (1).

$$\mathbf{x}_i = d_i \pi_i^{-1}(\mathbf{u}_i, \mathbf{K}_i). \tag{1}$$

We assign a random depth value d_i for $\mathbf{x}_i$, and the plane normal is defined in spherical coordinates

$$\mathbf{n}_i = \begin{bmatrix} cos\theta sin\phi \\ sin\theta sin\phi \\ cos\phi \end{bmatrix}. \tag{2}$$

where $\theta \in [0, 360°]$ is the angle between the plane's normal and the x axis. $\phi \in [0, 180°]$ is the angle between $\mathbf{n}_i$ and the z axis.

After initializing the random candidate plane to each pixel, we propagate initial guesses to neighboring pixels and add random perturbations. Here, we aim to find the support plane with the minimal aggregated matching cost for

each pixel. We first map a pixel to other view to find its corresponding patch using the assigned plane parameter. Then the matching cost is evaluated between two corresponding patches.

More specifically, for an input image I_i, each pixel $\mathbf{u}_i$ is mapped to a random plane and then is rendered onto the other view I_j^i. We use homography to perform planar mapping [23] defined in Eq. (3) with the plane parameters. To find the corresponding pixel in the other image I_j^i, we apply the projection function and transformation homography $\mathbf{H}_{ji}$ to each pixel's homogeneous coordinates $\mathbf{u_i}(u_i, v_i, 1)$ as Eq. (4). To obtain the direct dense matching on wide-angle images, we use the double sphere camera model given in Eqs. (3) and (4).

$$\mathbf{H}_{ji} = \mathbf{R}_j^T \mathbf{R}_i + \frac{1}{\mathbf{n}_i^T \mathbf{x}_i} \mathbf{R}_j^T (\mathbf{C}_j - \mathbf{C}_i) \mathbf{n}_i^T \tag{3}$$

$$[u_j^i, v_j^i, 1] = \pi_j \mathbf{H}_{ji} \pi_i^{-1} [u_i, v_i, 1] \tag{4}$$

Using the above equation, we locate the corresponding patch on I_j. We check the image variance between the corresponding patches centered at the given pixel. For a pixel $\mathbf{u}(u, v)$ in I_i, we select a correlation window $\mathcal{W}$ centered on that pixel position (u, v) and warp all the pixels in $\mathcal{W}$ to reference image I_j in order to find its corresponding patch I_j^i using Eq. 4. To evaluate image variance, we compute negative zero mean normalized cross correlation (ZNCC) over the window $\mathcal{W}$. A negative ZNCC between two corresponding patches at a given pixel position can be written as Eq. (5).

$$\mathcal{M}(\mathbf{u}, f) = \frac{-\sum\limits_{(x,y)\in\mathcal{W}} \{I_i(x,y) - \overline{I_i}(x,y)\}\{I_j^i(x,y) - \overline{I_j^i}(x,y)\}}{\sqrt{\sum\limits_{(x,y)\in\mathcal{W}} \{I_i(x,y) - \overline{I_i}(x,y)\}^2 \sum\limits_{(x,y)\in\mathcal{W}} \{I_j^i(x,y) - \overline{I_j^i}(x,y)\}^2}} \tag{5}$$

We traverse the given image in a row-wise order and optimize the parameters for each pixel. Then we perform propagation and random refinement. This optimization process is performed in many iterations for a single image. In the odd-numbered iterations, we start at the top left corner of the image and end at the bottom right corner. In the even-numbered iterations, we start at the bottom right corner until we reach the top left corner.

We examine whether the plane parameters of four neighboring pixels are better choices for the given pixel in spatial propagation. Here, we take the fact that neighboring pixels are more likely to lie on the same plane. In an odd-numbered iteration, we check the left pixel and the top pixel in spatial propagation. In an even-numbered iteration, we check the right pixel and the bottom pixel. Let the current pixel denoted by $\mathbf{p}$ and its plane denoted by f_p. We assign its neighboring pixel $\mathbf{q}$'s plane f_q to $\mathbf{p}$. We check the condition $\mathcal{M}(\mathbf{p}, f_q) < \mathcal{M}(\mathbf{p}, f_p)$, and if it holds, update $\mathbf{p}$ with $f_p = f_q$.

The random refinement searches for the plane parameters exhibiting a smaller matching cost. In this step, we alter the plane parameters within a wide range in the early stage. It is reasonable if the current parameters are completely

wrong. Then we reduce the search region, so the planes are close to the optimal one. Let the plane at point $\mathbf{p}$ be defined by the depth value d and the normal vector $\mathbf{n}$. We set the maximum change for the plane's depth Δd_{max} and the normal $\Delta\mathbf{n}(\Delta\theta_{max}, \Delta\phi_{max})$. Then we randomly choose values Δd from $[-\Delta d_{max}, \Delta d_{max}]$, and $\Delta\mathbf{n}(\Delta\theta, \Delta\phi)$. We can generate a new plane by adding those random values to $\mathbf{p}$'s plane $f'_p = [d + \Delta d, \mathbf{n} + \Delta\mathbf{n}]$. In our work, the variation range is set $\Delta d_{max} = \frac{1}{4}(d_{max} - d_{min})$, $\Delta\theta \in [0, 180°]$, $\Delta\phi \in [0, 90°]$. If the condition $\mathcal{M}(\mathbf{p}, f'_p) < \mathcal{M}(\mathbf{p}, f_p)$ holds, we update the plane in $\mathbf{p}$.

We perform the random refinement for a few iterations (e.g. 4) at each pixel and decrease the variation range by half in each iteration. This strategy effectively narrows the search space. It is essential to search a wide range in the early optimization stages, for there might be a false plane assignment. In later iterations, a compact variation range leads to precise exploration and allows accurate depth values for points on a smooth surface.

3 Experimental Results

We implemented our patch-based stereo algorithm using CUDA. Our algorithm can run effectively both on Windows and Linux. The performance of our algorithm was evaluated on a PC with NVIDIA GeForce GTX 2080Ti and 32GB main memory. We perform quantitative and qualitative comparisons against the state of the art fisheye stereo matching methods on public datasets. We perform the evaluations on two datasets: the TUM VI Benchmark [24] and the Oxford RobotCar dataset [25–27].

3.1 The TUM VI Benchmark

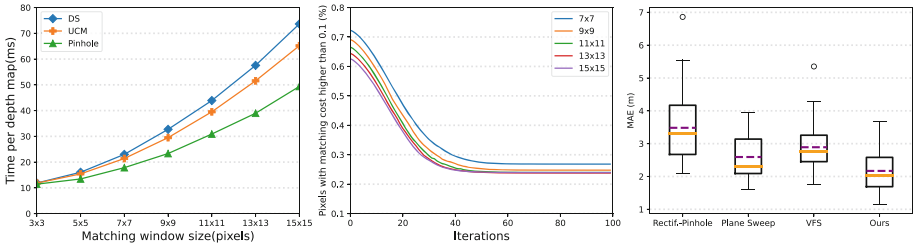

Fig. 1. Left: The average computational time per depth map under three camera models and various matching window. Middle: The pixel percentage of matching cost (<0.1) in terms of the number of iterations under various matching window. Right: MAE of four methods for the Oxford RobotCar dataset. The orange lines indicate the median values of the depth difference between the estimation and the LiDAR data. The purple dash lines highlight the mean values.

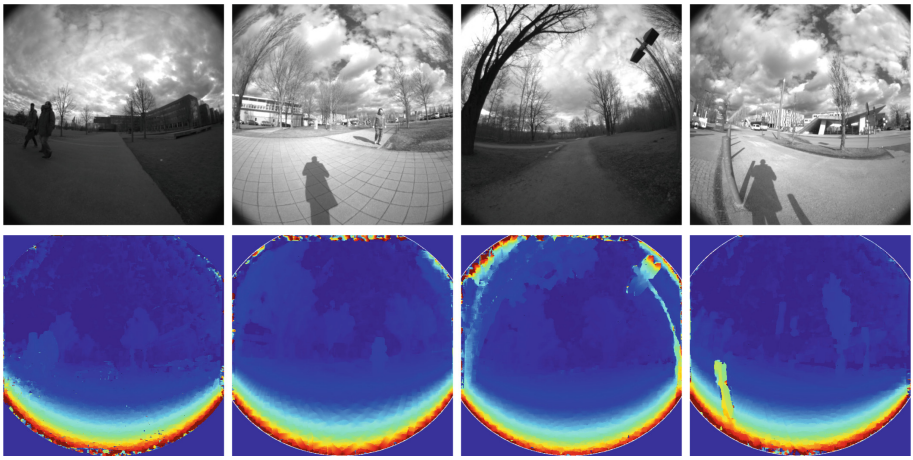

Fig. 2. Top Row: Input ultra-wide-angle images; Bottom Row: The corresponding ultra-wide-angle depth maps generated using our algorithm using double sphere camera model. Depth maps of the first and second columns are computed with the double sphere camera model, and the third and fourth columns use the unified camera model.

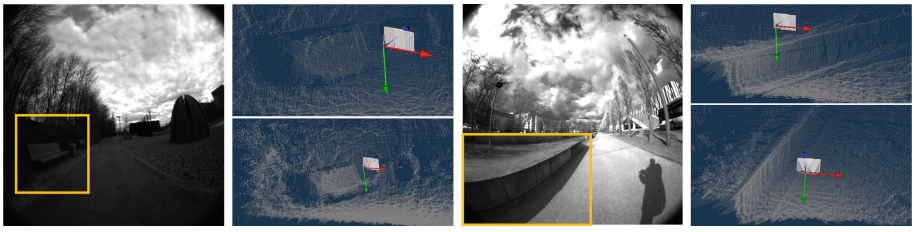

Fig. 3. The first and third images are input images, and the second and fourth images show the 3D point clouds constructed from our depth maps. Pay attention to the highlighted regions.

We tested our algorithm using the TUM VI Benchmark [24]. The benchmark contains an ultra-wide-angle stereo camera pair and provides calibration sequences. We first perform calibration to obtain the cameras intrinsics and extrinsics using Kalibr calibration toolbox [28]. In this experiment, the images have 512×512 pixels and 195° FOV. The matching window has 9×9 pixels. A larger window results in fewer noises (errors) in the resulting depth map. We first generate the depth map for one image and then take the output depth map as the initial guess for another. This greatly reduces the computational costs compared with random initialization for both images.

Figure 1-(Left) shows the computational time of generating a depth map under various matching window sizes. We implemented our patch-based stereo matching algorithm under three camera models: double sphere (DS), unified

camera model (UCM) and pinhole camera model. Note that we refer to the algorithm in [22] for rectified fisheye image under the pinhole camera model. In Fig. 1-(Middle), we also evaluated the matching cost in terms of the number of iterations. A larger matching window has a lower matching cost while taking more computational time.

In our experiment, the first image takes 60–80 iterations, while the second only takes 5–10 iterations. In addition, the depth result in successive frames can also be used as the initial guess for acceleration. Figure 2 show our experimental results. Our algorithm can generate continuous depth maps with clear object boundaries. Note that those images were taken in various lighting conditions. Our algorithm is less sensitive to illumination changes. In Fig. 3, we also show a few dense point clouds constructed from our depth maps. Moreover, our algorithm can work for any camera projection model, as long as it has a closed-form inverse. We tested our algorithm using the unified camera model. The third and fourth columns of Fig. 2 show some promising depth maps.

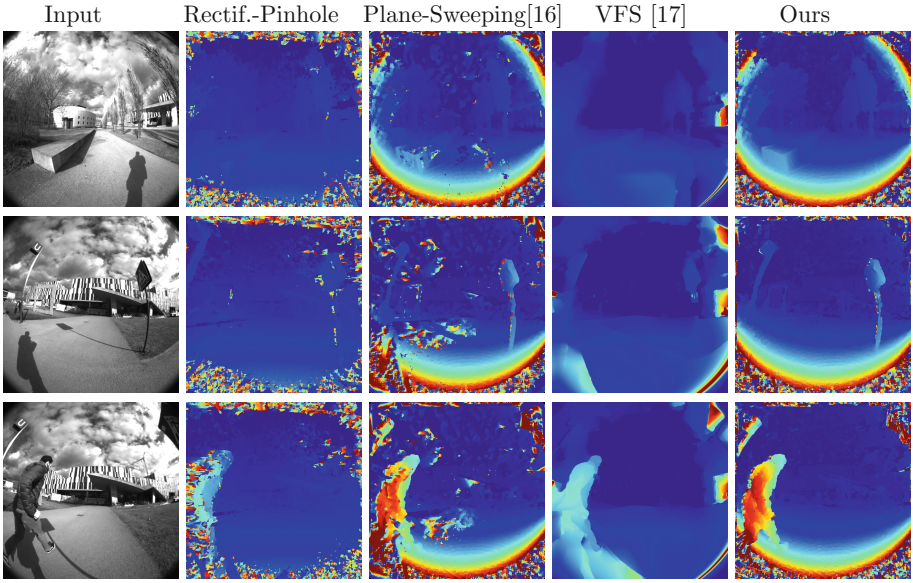

Fig. 4. Comparisons under the TUM VI dataset. Each row shows from left to right: input image, patch match stereo under the pinhole camera model [22], plane-sweeping [16], VFS [17] and our algorithm.

3.2 Comparisons

In Fig. 4, we compare our algorithm against the related work, including rectified wide-angle images under the pinhole camera model [22] (2nd column),

plane-sweeping stereo[1] [16] (3rd column), variational fisheye stereo (VFS)[2] [17] (fourth column). Plane-sweeping stereo and VFS directly perform stereo matching on wide-angle images. Plane-sweeping stereo searches for depth value from pre-defined hypotheses, and VFS searches for the correspondence along the epipolar curve using a trajectory field induced by camera motion. Note that the Kannala-Brandt camera model was used in VFS [29]. The last column in Fig. 4 shows the results using our algorithm. In these comparisons, the number of candidate planes is 256 in plane-sweeping. The number of iterations in the patch matching algorithm is 60. The matching window is 11×11. The computational performance and comparison are shown in Table 1.

Patch matching using the rectified images under the pinhole camera model exhibits clear noises near image boundaries. These boundary areas have to be cropped and therefore significantly reduce the valid coverage. Plane sweeping also generates a large number of inaccurate (noisy) depth values in the resulting depth maps. Besides the comparison results shown in Fig. 4, Fig. 5 shows another comparison against VFS. VFS generates depth maps with fewer noises, but the objects near the boundary are blended with the background. Our algorithm generates high-quality depth maps with clear and sharp object boundaries. Due to a large matching window, the objects may expand slightly along their boundaries.

We also perform quantitative comparison in terms of matching accuracy. For each pixel $\mathbf{u}$ in image I_i, we project the pixel into the world space $\mathbf{x}$ using the obtained depth d_i. Then, we map the pixel $\mathbf{u}$ onto the other image I_j to get its corresponding point $\mathbf{u}_j^i$, and project the point $\mathbf{u}_j^i$ into the space $\mathbf{x}'$ according to the depth map of I_j. Then we check if the condition $|\mathbf{x} - \mathbf{x}'| > \epsilon$ holds. If it holds, the depth value at $\mathbf{u}$ is considered invalid. We set $\epsilon = 2$. Table 1 shows the percentage of pixels that have depth errors less than $\epsilon = 2$. We also evaluate the matching cost in Eq. 5 for the resulting depth map. Our algorithm outperforms other algorithms in terms of depth error and matching cost (see Table 1).

Table 1. Performance and accuracy comparison

–	*TUM VI*			*Oxford RobotCar*	
	Time (ms)	Depth error <2(%)	Matching cost <0.2(%)	MAE	σ
Rectif.	28.5	63.12	85.88	3.483	2.923
Plane sweeping [16]	38.6	74.59	90.59	2.593	2.933
VFS [17]	207.8	80.92	40.68	2.962	2.763
Ours	39.4	83.56	92.49	2.169	2.552

[1] https://www.cvg.ethz.ch/research/planeSweepLib/.
[2] https://github.com/menandro/vfs.

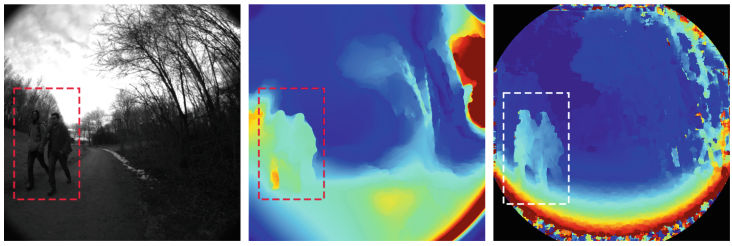

Fig. 5. Left: input images; Middle: depth maps generated using VFS; Right: our results. Pay attention to the rectangular areas. Our algorithm shows a clear object boundary compared with VFS.

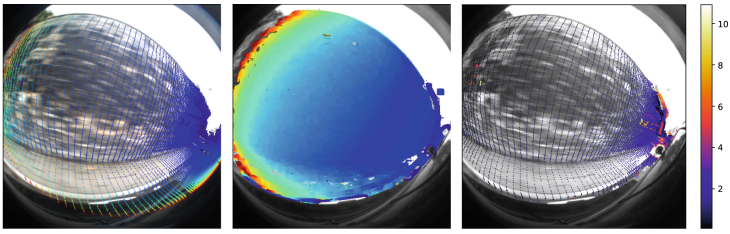

Fig. 6. Left: input image with LiDAR data projected; Middle: The resulting depth map computed by our algorithm. Right: The depth error per pixel where the LiDAR data are available.

3.3 The Oxford RobotCar Dataset

We further evaluated our algorithm using the Oxford RobotCar dataset with LiDAR, GPS and INS ground truth. The wide-angle image sequences are collected from three wide-angle cameras mounted on the left, the right and the rear. We use the left-mounted global shutter camera images with a 180° FOV. We calibrate the camera with the Kalibr toolbox to obtain the intrinsic parameters. For the extrinsic of each image, we apply the unified camera model to ORB-SLAM2 [30] to get the keyframe poses. Then we compute the depth maps

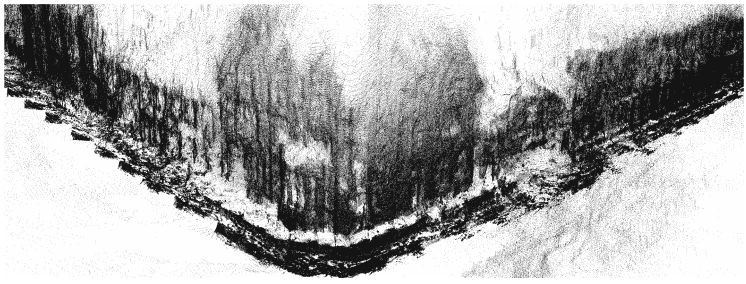

Fig. 7. The point cloud generated using our algorithm.

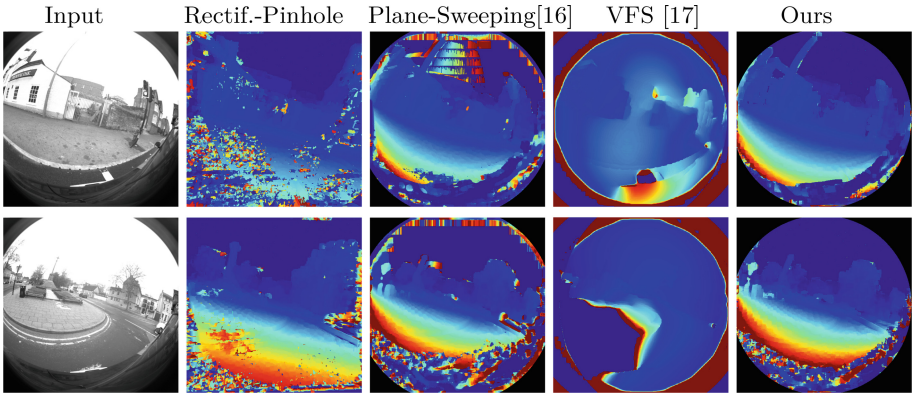

Fig. 8. Depth maps generated using four methods. The patch match stereo on rectified wide-angle images generated noise(inaccurate depth values), and the plane sweeping fails at the background sky area. VFS does not work well on images with significant displacement. Our method can generate high-quality depth maps with clear object boundaries.

for these keyframes. All the stereo matching algorithms are performed for two consecutive images. We compare our algorithm with point clouds scanned using LiDAR, as shown in Fig. 6. We observe that the depth maps generated using our algorithm are close to the depth resulted from the LiDAR. Figure 7 shows the point clouds generated by the proposed algorithm.

We also performed a comparison against other algorithms using the Oxford Radar RobotCar Dataset. The comparison results are given in Fig. 8. The matching window is 9×9 and 11×11 for evaluating the matching cost. The maximum number of iterations for patch match stereo is 60. The number of candidate planes for plane sweeping is 256.

We also compare the depth maps using the Velodyne HDL-32E LiDAR scans on the dataset. We evaluate the mean absolute error (MAE) on every depth map for the pixels whose depth is less than 20 m. We calculate the scale factor for every depth map. We take 50 keyframes and evaluate the dense stereo matching algorithms. The comparison result is shown in Table 1 and Fig. 1-(Right). The experimental results indicate that our algorithm has the best performance in comparison.

4 Conclusion

In this paper, we demonstrate that patch-based stereo matching can be directly applied to ultra-wide-angle images. Our algorithm can directly estimate depth values at each pixel for the given ultra-wide-angle images and generates high-quality depth maps with clear object boundaries. We further accelerate our algorithm using GPUs.

Our work has some limitations. For texture-less or shining surfaces, there exhibit false matches due to insufficient local evidence. This may cause obvious noises in the resulting depth maps.

In future, we would like to investigate the patch-based stereo matching algorithm in the stereoscopic 360 camera rig in order to generate a full range of depth maps for the surrounding environment. In addition, we would like to consider mounting stereo ultra-wide-angle cameras onto our mobile robot platform. Robot path planning is expected to greatly benefit from the resulting depth maps.

Acknowledgement. This work is supported by the National Key R&D Program of China under grant 2021ZD0114501.

References

1. Tao, Y., Shen, Y., Sheng, B., Li, P., Lau, R.W.H.: Video decolorization using visual proximity coherence optimization. IEEE Trans. Cybern. **48**(5), 1406–1419 (2018)
2. Zhang, B., Sheng, B., Li, P., Lee, T.Y.: Depth of field rendering using multilayer-neighborhood optimization. IEEE Trans. Vis. Comput. Graph. **26**(8), 2546–2559 (2020)
3. Guo, H., Sheng, B., Li, P., Chen, C.L.P.: Multiview high dynamic range image synthesis using fuzzy broad learning system. IEEE Trans. Cybern. **51**(5), 2735–2747 (2021)
4. Courbon, J., Mezouar, Y., Eckt, L., Martinet, P.: A generic fisheye camera model for robotic applications. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1683–1688 (2007)
5. Gu, Z., Liu, H., Zhang, G.: Real-time indoor localization of service robots using fisheye camera and laser pointers. In: IEEE International Conference on Robotics and Biomimetics, pp. 1410–1414 (2014)
6. Zhang, X., Zhao, Y., Mitchell, N., Li, W.: A new 360 camera design for multi format VR experiences. In: IEEE Conference on Virtual Reality and 3D User Interfaces, pp. 1273–1274 (2019)
7. Häne, C., et al.: 3D visual perception for self-driving cars using a multi-camera system: calibration, mapping, localization, and obstacle detection. Image Vis. Comput. **68**, 14–27 (2017)
8. Kumar, V.R., Klingner, M., Yogamani, S., Milz, S., Fingscheidt, T., Mader, P.: SynDistNet: self-supervised monocular fisheye camera distance estimation synergized with semantic segmentation for autonomous driving. In: IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 61–71 (2021)
9. Lee, S., Kim, G.J., Choi, S.: Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. IEEE Trans. Vis. Comput. Graph. **15**(3), 453–464 (2009)
10. Wen, Y., et al.: Structure-aware motion deblurring using multi-adversarial optimized CycleGAN. IEEE Trans. Image Process. **30**, 6142–6155 (2021)
11. Liu, S., Guo, P., Feng, L., Yang, A.: Accurate and robust monocular SLAM with omnidirectional cameras. Sensors **19**(20), 4494 (2019)
12. Wang, Y., et al.: CubemapSLAM: a piecewise-pinhole monocular fisheye SLAM system. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) ACCV 2018. LNCS, vol. 11366, pp. 34–49. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20876-9_3

13. Shah, S., Aggarwal, J.: Depth estimation using stereo fish-eye lenses. In: IEEE International Conference on Image Processing, vol. 2, pp. 740–744 (1994)
14. Li, S.: Real-time spherical stereo. In: IEEE International Conference on Pattern Recognition, vol. 3, pp. 1046–1049 (2006)
15. Li, J., Xu, W., Zhang, Z., Zhang, M., Wang, Z.: Fisheye image rectification for efficient large-scale stereo. In: International Conference on Systems and Informatics, pp. 881–885 (2016)
16. Häne, C., Heng, L., Lee, G.H., Sizov, A., Pollefeys, M.: Real-time direct dense matching on fisheye images using plane-sweeping stereo. In: IEEE International Conference on 3D Vision, p. 57–64 (2014)
17. Roxas, M., Oishi, T.: Variational fisheye stereo. IEEE Robot. Autom. Lett. **5**(2), 1303–1310 (2020)
18. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: a randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
19. Sheng, B., Li, P., Gao, C., Ma, K.L.: Deep neural representation guided face sketch synthesis. IEEE Trans. Vis. Comput. Graph. **25**(12), 3216–3230 (2019)
20. Usenko, V., Demmel, N., Cremers, D.: The double sphere camera model. In: International Conference on 3D Vision, pp. 552–560 (2018)
21. Bleyer, M., Rhemann, C., Rother, C.: PatchMatch stereo-stereo matching with slanted support windows. In: British Machine Vision Conference, vol. 11, pp. 1–11 (2011)
22. Shen, S.: Accurate multiple view 3D reconstruction using patch-based stereo for large-scale scenes. IEEE Trans. Image Process. **22**(5), 1901–1914 (2013)
23. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press, Cambridge (2004)
24. Schubert, D., Goll, T., Demmel, N., Usenko, V., Stueckler, J., Cremers, D.: The TUM VI benchmark for evaluating visual-inertial odometry. In: International Conference on Intelligent Robots and Systems, pp. 1680–1687 (2018)
25. Barnes, D., Gadd, M., Murcutt, P., Newman, P., Posner, I.: The Oxford radar robotcar dataset: a radar extension to the Oxford robotcar dataset. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 6433–6438 (2020)
26. Maddern, W., Pascoe, G., Gadd, M., Barnes, D., Yeomans, B., Newman, P.: Real-time kinematic ground truth for the Oxford robotcar dataset. arXiv preprint arXiv: 2002.10152 (2020)
27. Maddern, W., Pascoe, G., Linegar, C., Newman, P.: 1 year, 1000km: the Oxford RobotCar dataset. The Int. J. Robot. Res. **36**(1), 3–15 (2017)
28. Furgale, P., Rehder, J., Siegwart, R.: Unified temporal and spatial calibration for multi-sensor systems. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1280–1286 (2013)
29. Kannala, J., Brandt, S.S.: A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. IEEE Trans. Pattern Anal. Mach. Intell. **28**(8), 1335–1340 (2006)
30. Mur-Artal, R., Tardós, J.D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans. Robot. **33**(5), 1255–1262 (2017)