2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Omnidirectional Dense SLAM for Back-to-back Fisheye Cameras

Weijian Xie[1,2] Guanyi Chu[2] Quanhao Qian[2] Yihao Yu[2] Shangjin Zhai[2]
Danpeng Chen[1,3] Nan Wang[2] Hujun Bao[1] Guofeng Zhang[1*]
[1]State Key Lab of CAD&CG, Zhejiang University [2]SenseTime Research [3]Tetras.AI

***Abstract*— We propose a real-time visual-inertial dense SLAM system that utilizes the online data streams from back-to-back dual fisheye cameras setup, providing 360° coverage of the environment. Firstly, we employ a sliding-window-based front-end to estimate real-time poses from the binocular fisheye images and IMU data. Then, we implement a lightweight panoramic depth completion network based on multi-basis depth representation. The network takes panoramic images (obtained by stitching dual-fisheye images with extrinsics and intrinsic parameters) and sparse depths (generated by the front-end local tracking) as input and predicts multiple depth bases along with corresponding confidence as output. The final dense depth is the linear combination of the multiple depth bases. Thanks to the multi-basis depth representation, we can continuously optimize the 360° depth with the traditional optimizer to achieve higher global consistency in depth. We conducted experiments on both simulated and real-world datasets to evaluate our method. The results demonstrate that the proposed method outperforms SoTA methods in terms of depth prediction and 3D reconstruction. In addition, we develop a demo that can run on a mobile to demonstrate the real-time capabilities of our method.**

I. INTRODUCTION

Real-time pose estimating and dense mapping in unknown environments is a fundamental challenge in robot navigation, autonomous driving, and AR/VR applications, especially on resource-constrained devices. Indeed, distance sensors like LiDAR can directly provide depth information, which is valuable for dense mapping [1]. However, these sensors face practical challenges such as low vertical resolution, size limitations, cost, and power consumption. Compared to distance sensor solutions, camera sensors still stand as a more versatile and cost-effective option. Among them, panoramic camera systems have gained significant attention and development in recent years because they can capture 360-degree scene information.

Various hardware solutions for capturing panoramic images have emerged [2]. The most commonly used multi-camera setups include the back-to-back binocular cameras [3] and four fisheye cameras [4]. 360-degree devices using back-to-back dual fisheye setups like Insta360[1], GoPro Max[2] have been successfully commercialized. Meanwhile, research based on panoramic images has also made significant progress, including depth prediction [5]–[7] and SLAM [8]–[10]. A natural idea is to combine the predicted 360° depth map with SLAM systems [11], [12], thereby achieving 3D geometric mapping. However, most previous work [11], [13]–[15] has mainly focused on post-processed panoramic images rather than online data streams. Applying these methods to online fisheye data streams remains challenging, especially on resource-constrained devices. [6], [12], [16] based on fisheye cameras has primarily concentrated on the hardware setups with four fisheye cameras. Unlike the four-camera fisheye setup, the back-to-back fisheye cameras capture a minimal overlap in the field of view, which increases the difficulty of fisheye-based depth prediction.

This paper proposes a novel dense VI-SLAM system based on online streams from a back-to-back dual-fisheye system. To the best of our knowledge, it is the first attempt to achieve

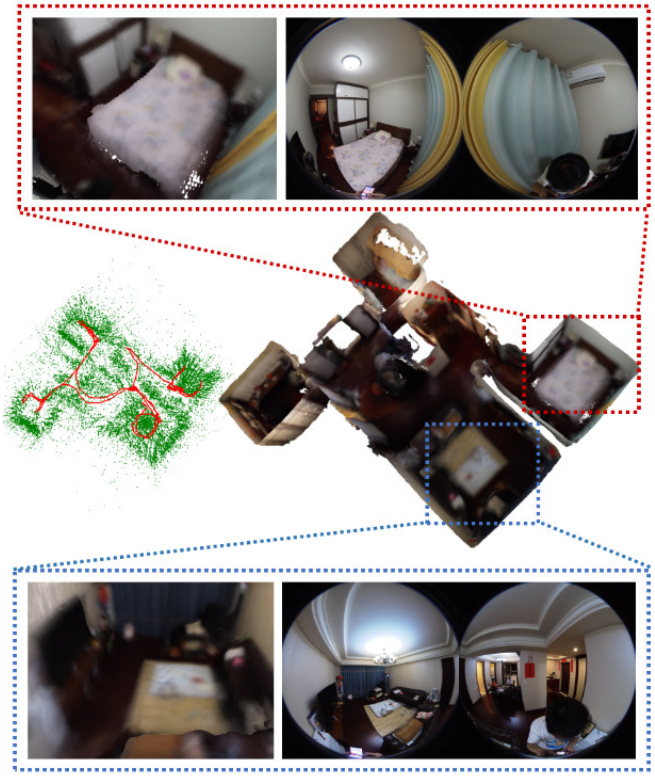


Fig. 1: Reconstruction of real-world scenes captured using Insta360 devices. The middle row displays the sparse point cloud, trajectory, and the reconstructed dense mesh. The top and bottom rows show details of the reconstructed mesh and corresponding views from the dual fisheye images, respectively.

*Corresponding author: Guofeng Zhang (zhangguofeng@zju.edu.cn)
This work was partially supported by NSF of China (No. 61932003).
[1]https://www.insta360.com/
[2]https://gopro.com/

this objective. Figure 1 shows the result in real-world of our method. Unlike [12], our front-end tracking does not rely on network-predicted depth, ensuring real-time performance. In contrast to [11] which uses post-processed panoramic images as input, our method accepts online stereo fisheye images, truly meeting the demands of real-time pose estimation and dense mapping. Utilizing multi-basis depth representation with back-to-back fisheye devices presents unique challenges compared to [17]. To ensure both efficiency and mapping accuracy, we developed a framework tailored for tracking on stereo fisheye images and mapping on panoramic images. For this purpose, we customize a VIO system and a fast image-stitching algorithm. Additionally, we refine the network training and outlier removal strategy to mitigate the distortion effects on depth prediction and optimization. The contributions of this paper can be summarized as follows:

- We present a novel real-time dense visual-inertial SLAM system based on a back-to-back dual-fisheye configuration, which can run in real-time on mobile devices. Our system takes online inputs from binocular fisheye cameras and IMU data.
- We propose a lightweight depth completion network tailored for panoramic SLAM systems. The network takes sparse points (generated by SLAM system) and panoramic images as inputs, and predicts depth bases and confidence. The final depth information is obtained through a linear combination of the depth bases.
- By leveraging the representation of multiple depth bases, we incorporate weights of depth bases into the traditional optimization of SLAM, thereby achieving continuous refinement of panoramic depth.

II. Related Work

A. Depth Prediction

Most panoramic depth prediction methods [18]–[21] rely on post-processed ERP (Equirectangular projection) images, highlighting the importance of addressing spherical distortion in such images. To tackle the problem of uneven distortion in ERP images, these efforts primarily focus on improving the network's receptive field. To combat uneven distortion, efforts have centered on enhancing the network's receptive field. Some methods [22], [23] take ERP images with CP(Cubemap Projection) images as input, leveraging the strengths of both: ERP images are continuous but uneven, while CP images are uniform but discontinuous. BiFuse [22] recovers depth from both ERP and CP images using a dual-data stream approach, whereas Unifuse [23] only employs a dual-data stream during the encoding phase. Both methods have yielded promising results. PanoFormer [13] and HRDFuse [14] use Tangent Patches as network inputs to address distortion effects, yielding excellent results. However, achieving precise depth remains challenging due to the inherently ill-posed nature of monocular depth prediction.

OmniMVS [7] and OmniVidar [6] leverage multi-view geometry to predict depth, but such methods are only suitable for four-camera setups with overlap. While depth completion has shown promise with small FOV cameras, limited research has focused on panoramic images. [24] relies on a very high density of sparse points, which may not suit the extremely sparse depth generated by a SLAM system.

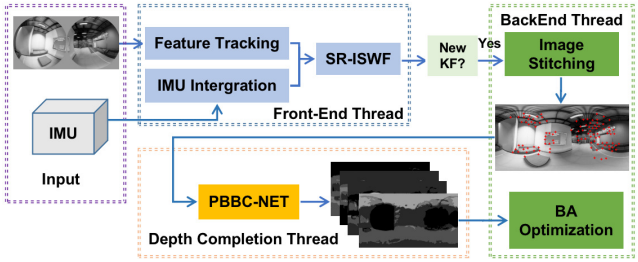


Fig. 2: System Overview.

B. SLAM

Small field of view (FOV) SLAM systems [25]–[30], whether based on a single camera or multiple cameras, whether integrating IMU or other sensors, have mature solutions available. Most SLAM systems designed for panoramic camera setups primarily rely on Visual Odometry (VO) methods, such as OpenVSLAM [27], ORB-SLAM3 [25], 360VO [9], and ROVO [10]. VISLAM solutions based on panoramic systems are highly dependent on hardware setup, and there is limited work combining them with depth prediction. ROVINS [31] is tailored for four-eye cameras, while LF-VIO [32] is designed for Panoramic Annular Lens cameras. Omni-SLAM [12] merges OmniMVS [7] and ROVO [10] for dense mapping but is unsuitable for camera setups lacking overlap, like back-to-back cameras. Moreover, its localization relies on predicted depth, limiting the system's efficiency. [11] combines a depth completion network with ORB-SLAM. However, [11] relies on post-processed panoramic images, and the predicted depth is not continuously optimized. [17], [33], [34] both demonstrate the significance of continuously optimizing the depth predicted by the network for improving the global consistency of depth.

III. Methods

The system overview of our method shows in Figure 2. The system takes two fisheye cameras and IMU as inputs. The front-end tracking module performs real-time pose estimation based on fisheye images and IMUs. The keyframes and local point clouds generated by front-end module are then used for the back-end dense mapping. The back-end module comprises three main steps. Firstly, we stitch the two fisheye images into an ERP image based on the extrinsic and intrinsic parameters. Then, we use this stitched ERP image and the local sparse point cloud as inputs to the network. The network predicts multiple depth bases and confidence. Finally, we incorporate the weight of depth bases and confidences into the window optimization, further enhancing depth consistency.

Notation Here, we predefined certain notations and symbols for the sake of clarity. Rotation and translation are denoted by R and t respectively. We refer to the label on the left side of the symbol as the transformation between two coordinates, and the label on the right side of the symbol as the image index. The coordinate we use include two fisheye (F_l and F_r), world (W), body (B), virtual spherical space (S). For instance, ${}^{W}_{B}R_i$ and ${}^{W}_{B}t_i$ are the rotation and translation from the body to the world of image i. Specifically, ${}^{S}_{B}t$ equal to $({}^{F_l}_{B}t + {}^{F_r}_{B}t)/2$. ${}^{S}_{B}R$ is manually specified to ensure that the left and right halves of the ERP image correspond to two fisheye images. π_S and π_F are the projection function to project 3D point to the ERP image and fisheye images respectively. X denote the 3D point in space, while x denote the 2D point in image. For spherical space, ${}^{S}X = \dot{d}\psi(\theta, \phi)$, where θ and ϕ are the longitude and latitude in spherical coordinate. $\dot{d}$ is the distance from ${}^{S}X$ to the center of sphere, which is equal to $||{}^{S}X||$.

A. Front-End

The front-end of our system boosts efficiency by integrating SR-ISWF [30], a sliding-window design for VIO, along with a square-root inverse filter to fuse all measurements. The specific implementation of our VIO is the same as the binocular mode in [35]. For the new binocular fisheye input, we employ a FAST corner detector to extract corner points and use the KLT method for optical flow tracking in each fisheye image. After visual tracking, the tracked features are first transformed from the fisheye image coordinate to the normalized image coordinate. Afterward, we can add these visual observations to the filter of VIO, just like handling a regular camera. In particular, considering the significant distortion in fisheye images, we assign different weights to each tracked sparse point based on its position. We aim to reduce the weight of points in regions with significant distortion. The weight is defined as follows:

$$w_f(u,v) = \begin{bmatrix} \left|\pi_F^{-1}(u+1,v+1)[0] - \pi_F^{-1}(u,v)[0]\right|^{-1} \\ \left|\pi_F^{-1}(u+1,v+1)[1] - \pi_F^{-1}(u,v)[1]\right|^{-1} \end{bmatrix}, \tag{1}$$

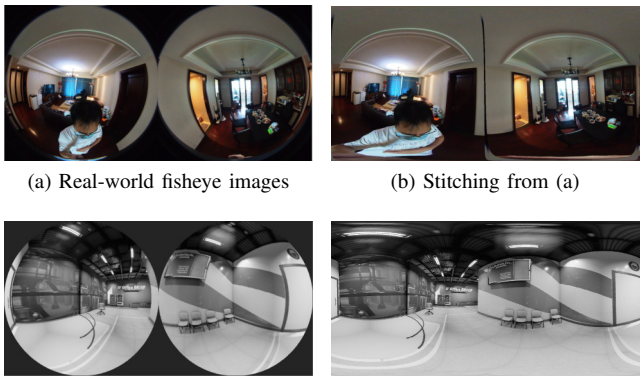
where $\pi_F^{-1}(\cdot)$ up-projects point from image to normalized image coordinate. The weights are used for both outlier rejection and state updates.

B. Depth Completion

Image Stitching Before using the network to complete the depth, we need to generate an ERP image from fisheye images. To ensure efficiency, we did not implement a complete image stitching algorithm as [36]. In contrast, we directly employ the extrinsic and intrinsic parameters to stitch the two fisheye camera images into a panoramic image. For each point x_S in the ERP image, the corresponding point in fisheye images can be calculated as follows:

$$\begin{aligned} x_{F_l} &= \pi_{F_l}({}^{F_l}_{S}R\pi_S^{-1}(x_S) + {}^{F_l}_{S}t), x_S[0] < width/2 \\ x_{F_r} &= \pi_{F_r}({}^{F_r}_{S}R\pi_S^{-1}(x_S) + {}^{F_r}_{S}t), x_S[0] >= width/2 \end{aligned} \tag{2}$$

Equation 2 establishes the mapping relationship between the pixel points on the fisheye image and the pixel points on the ERP image. By employing the interpolation method, we can easily accomplish an efficient transformation from the fisheye image to the ERP image. The mapping relationship depends solely on the extrinsic and intrinsic parameters, so it only needs to be computed once when the system is initiated. This approach allows for a rapid fusion of the two fisheye images into a panoramic image. For a 1920×960 resolution, our method takes only 4ms, while the method proposed in [36], even with multi-threading, requires around 70ms. The stitching results are shown in Figure 3. It can be observed that due to hardware limitations (such as the misalignment of the optical centers of the two fisheye cameras), using image stitching based on external and internal parameters for images captured by actual devices may result in strange seams along the boundaries. This issue can be easily resolved by masking out the predicted depth in that region, and the user's head can also be removed in this way.



(a) Real-world fisheye images (b) Stitching from (a)

(c) Synthetic fisheye images (d) Stitching from (c)

Fig. 3: Image stitching by extrinsic and intrinsic parameters.

Network Our network utilizes the same architecture as described in [17], which is a symmetric Res-UNet shown in Figure 4. Our network (PBBC-Net) takes an ERP image (generated by image stitching) and sparse depth points (generated by Front-End tracking) as input and outputs multi-basis depths and confidence. ResNet34 [37] is used as the encoder. The final depth is estimated by $D = \sum_k^n w_k^d B_k$, where w_k^d is the weight of basis B_k and n is the num of depth bases. For keyframe *i*, we project sparse map points generated by VIO to spherical coordinates as follows:

$${}^{S}X_i = {}^{S}_{B}R({}^{B}_{W}R_i\,{}^{W}X + {}^{B}_{W}t_i) + {}^{S}_{B}t. \tag{3}$$

Compared to [17], the primary changes include a customized loss function and data augmentation strategies designed specifically for panoramic images. We retained depth loss L_d, confidence loss L_c, and base balance loss L_b. The confidence loss is defined as follows:

$$L_c = ||(D_{gt} - D_{pred}) \circ C||_1 + w_0 \left\| \frac{1}{C+1} \right\|_1, \tag{4}$$

where *C* is the predict confidence, w_0 is the hyper-parameter that penalizes confidence *C* close to 0, which is set to 0.1 in this paper.

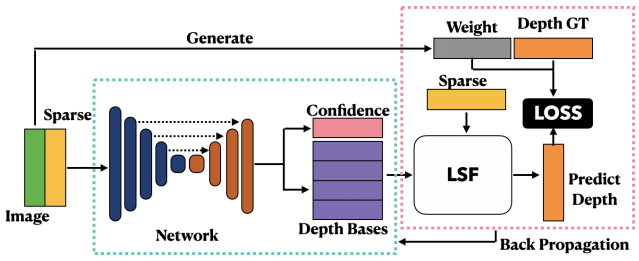


Fig. 4: Network.

The base balance loss is defined as follows:

$$L_b = \log(\lambda_{max}(B^T B)) - \log(\lambda_{min}(B^T B)), \tag{5}$$

where $B^T B$ is the model matrix of the least-squares problem composed of sparse depth observation. The max and min eigenvalue of matrix $B^T B$ are λ_{max} and λ_{min} respectively.

The depth loss is defined as follows:

$$L_d = ||(D_{gt} - D_{pred})||_1. \tag{6}$$

Specifically, to overcome the issue of spherical distortion in ERP image, similar to [11], we generate a weight matrix as follows:

$$W_E(u,v) = W_E(\psi(\theta,\phi)) = \int_{\theta}^{\theta+1} d_\theta \int_{\phi}^{\phi+1} R^2 cos\phi d\phi. \tag{7}$$

Where θ, ϕ is the spherical coordinates of the pixel corresponding to u, v.

To get better performance, inspired by [38], we introduced a virtual normal loss, defined as follows:

$$L_{vn} = \frac{1}{N_n} \sum_{i=0}^{N_n} \left\| n_i^{pred} - n_i^{gt} \right\|_1, \tag{8}$$

where N_n is the number of sampling points, n_i^{pred} and n_i^{gt} are the predicted virtual normal and ground truth virtual normal respectively. For more details about virtual normal please refer to [38].

So the whole loss function is defined as follows:

$$L = \mu_d W_E L_d + \mu_c W_E L_c + \mu_b L_b + \mu_{vn} L_{vn}, \tag{9}$$

where μ_d, μ_c, μ_b, μ_{vn} are the hyperparameters that fuse each loss, which are 1, 0.1, 0.01, 0.5 in this paper respectively. When applying padding to the convolutional layers, we use circular padding on both sides of the features horizontally since both sides of the panoramic image are connected.

Besides, we performed data augmentation on the training data during the training process, including horizontal flip, random brightness contrast, random gamma, and random Gaussian noise. Taking into account the characteristics of the panoramic images, geometrical augmentation is performed by mapping the panoramic image to the unit sphere. We first rotate the pixels of the image randomly ([-180°,180°] around the y-axis and [-11.25°,-11.25°] around the z-axis) on the unit sphere, then map the pixels back to the panoramic image.

C. Depth Optimization

Depth Factors Thanks to the multi-basis depth representation, we incorporate the predicted depth bases and their weights into the traditional optimizer. Similar to [17], we employ the relative depth factor, sparse depth factor, and prior weight factor for depth optimization. We use the map points from front-end module to generate sparse depth factors, and use sampled 2D points on ERP images(similar to the method in [28]) to generate relative depth factors. The prior weight factor is introduced to enhance optimization robustness.

For each 3D point ${}^W X$ generated by VIO, its corresponding position in ith frame is:

$${}^S X_i = {}^S_B R({}^B_W R_i \, {}^W X + {}^B_W t_i) + {}^S_B t. \tag{10}$$

Then, the sparse depth loss can be defined as follows:

$$E_{D_i} = \left\| {}^S X_i \right\| - \sum_k^n (w_i^k B_i^k(\pi_S({}^S X_i))). \tag{11}$$

For each 2d point in ith ERP image ${}^p x_i$, we can get its 3D point by:

$${}^S X_i = \pi_S^{-1}({}^p x_i) D({}^p x_i) = \pi_S^{-1}({}^p x_i)(\sum_k^n w_i^k B_i^k({}^p x_i)). \tag{12}$$

Then, we can convert it to jth ERP image's spherical coordinate by

$${}^S X_j = {}^S_W R_j({}^W_S R_i \, {}^S X_i + {}^W_S t_i) + {}^S_W t_j. \tag{13}$$

So, the relative depth loss can be defined as follows:

$$E_{R_{ij}} = \left\| {}^S X_j \right\| - \sum_k^n (w_j^k B_j^k(\pi_S({}^S X_j))). \tag{14}$$

The definition of the prior loss is the same as in [17], as follows:

$$E_{P_i} = \hat{w}_i - w_i, \tag{15}$$

where $\hat{w}_i$ is the initial value of w_i before optimization.

Outlier Removal Compared to depth completion results from small FoV cameras, we have noticed that depth completion networks based on panoramic images are more sensitive to sparse depth noise, and the predicted depths also tend to have more significant noise. Therefore, in addition to confidence, we have employed some strategies for more rigorous noise filtering. When constructing sparse depth for frame *i*, we project the sparse 3D points ${}^S X$ generated by the front-end onto nearby keyframes *j* where predicted depths exist. We assess the sparse depth loss of ${}^S X$ on frame *j*, and if it exceeds the threshold (0.5 in this paper), the sparse point is discarded. Similarly, when generating relative depth factors, we also assess the relative depth error of sample points ${}^p x_i$ on frame *i* to their adjacent frames *j*. If the relative depth error surpasses a certain threshold (0.5 in this paper), the sample point is discarded. In addition to checking relative depth errors, we also verify the grayscale intensity values of the sample points. If the difference between the

grayscale intensity value of the sample point on the frame *i* and its counterpart on the frame *j* exceeds the threshold (56 in this paper), then the sample point will not be added into optimizer.

IV. Experiment

Implement Details We use a Nvidia Tesla V100 SXM2 GPU for training. We conduct experiments on a desktop PC with an Intel i7-8700 CPU and 24 GB RAM, and use an Nvidia GTX 1060 6GB GPU for inference. We first train our model without confidence loss for 50 epochs and then with confidence for 50 epochs. We employ the ADAMW optimizer with a dynamic learning rate adjustment. It begins at 1e-5 and warms up to 1e-3 over the first 100 steps, followed by a cosine annealing schedule that reduces it to 1e-5 for the remaining training duration. For each frame, we sample the sparse depths from the ground truth depth of FAST corners. We use 150 sparse points for training, consistent with the average number of map points contained in keyframes. To evaluate our method, we chose to quantitatively compare our work using the metrics previously used in the literature and metrics, including AbsRel, SqRel, RMSE, RMSLE (RMSE-log), $\delta(1.25, 1.25^2)$.

Synthetic Dataset Due to the absence of datasets containing sequential motion data with panoramic and IMU data for evaluating panoramic dense visual-inertial SLAM, we generated some synthetic data using AirSim [39]. AirSim has the capability to generate simulated images along with depth and IMUs from the Unreal Engine. We have generated a total of 5 dataset sequences, encompassing three room scenes with average depths of approximately 1-3 meters (room1), 2-4 meters (room2), and 3-5 meters (room3).

Depth Completion Analysis We evaluated our network on Matterport3D and Stanford2D3D and compared it to SoTA depth prediction methods including Bifuse [22], UniFuse [23], PanoFormer [13] and HDRFuse [14]. For comparison, we train our network on Matterport3d [40] and Standford2d-3d [41] independently. For Matterport3d, we follow the official split, while for Standford2d-3d we follow the split used by [23]. The result shows in Table I. Depth completion methods based on panoramic images are limited. [11] was evaluated only on the now-unavailable 360D dataset. [24] utilized a significantly larger number of sparse points (7%) compared to our method (less than 0.1%), rendering accuracy comparisons unfair. Neither [11] nor [24] have published their code, so a fair comparison with them cannot be made. Therefore, we present the results of BBC-Net* as a reference in Table I. BBC-Net* is a simple adaptation of BBCNet [17], a small FoV depth completion network, to the panoramic depth completion scenario. Specifically, We removed incompatible geometrical data augmentation and retrained the BBCNet on Matterport3d with 150 points.

From Table I, it is evident that our network exhibits a significant improvement in accuracy compared to SoTA depth prediction networks. Even the results of BBC-Net are notably superior to the SoTA methods. This demonstrates that the incorporation of sparse depth information plays a substantial role in enhancing depth accuracy. Additionally, it can be observed that the depth prediction accuracy of our network improves with the number of sparse points increases.

TABLE I: Quantitative comparison with the SOTA methods.

		$\delta 1$	$\delta 2$	AbsRel	SqRel	RMSE	RMSLE
Matterport3D	Bi [22]	0.845	0.932	0.2048	-	0.6259	-
	Uni [23]	0.890	0.962	0.1063	-	0.4941	-
	PF [13]	0.882	0.966	0.0904	0.0764	0.4470	0.1650
	HRD [14]	0.916	0.967	0.0967	0.0936	0.4433	0.1642
	Ours(150)	0.961	0.989	0.0504	0.0334	0.2972	0.0967
	Ours(650)	**0.983**	**0.995**	**0.0285**	**0.0176**	**0.2128**	**0.0678**
	BBCNet*	0.951	0.986	0.0613	0.0419	0.3254	0.1076
Stanford2D3D	Bi [22]	0.866	0.958	0.1209	-	0.4142	-
	Uni [23]	0.871	0.966	0.1114	-	0.3691	-
	PF [13]	0.881	0.962	0.1131	0.0723	0.3557	0.2454
	HRD [14]	0.914	0.980	0.0935	0.0508	0.3106	0.1422
	Ours(150)	0.966	0.990	0.0471	0.0396	0.2485	0.1000
	Ours(650)	**0.981**	**0.993**	**0.0278**	**0.0232**	**0.2056**	**0.0799**
	BBCNet*	0.951	0.987	0.0659	0.0433	0.2793	0.1157

TABLE II: The ablation result of network. The best result is shown in bold. BBCNet* is the baseline of our method. *filtered (5%)* stands for use confidence to filter out 5% point according to the confidence. *vnl loss* stands using virtual normal loss. *confidence* stands for using confidence loss. *weight* stands for using weight defined in Equation 7. *aug* stands for using data augment. We sample 150 FAST coreners as sparse depth for input.

	$\delta 1$	AbsRel	SqRel	RMSE	RMSLE
BBCNet*	0.951	0.0613	0.0419	0.3254	0.1076
full	0.961	0.0504	0.0334	0.2972	0.0967
filtered(5%)	**0.972**	**0.0431**	**0.0063**	**0.1283**	**0.0494**
w/o *vnl loss*	0.958	0.0524	0.0349	0.3036	0.0985
w/o *confidence*	0.960	0.0532	0.0346	0.2993	0.0984
w/o *weight*	0.960	0.0508	0.0338	0.2998	0.0976
w/o *aug*	0.960	0.0503	0.0352	0.3017	0.0972

Ablation Study To better illustrate the benefits of our various modifications on accuracy improvement, we conducted ablation experiments on both the network improvements and the depth-SLAM fusion improvements.

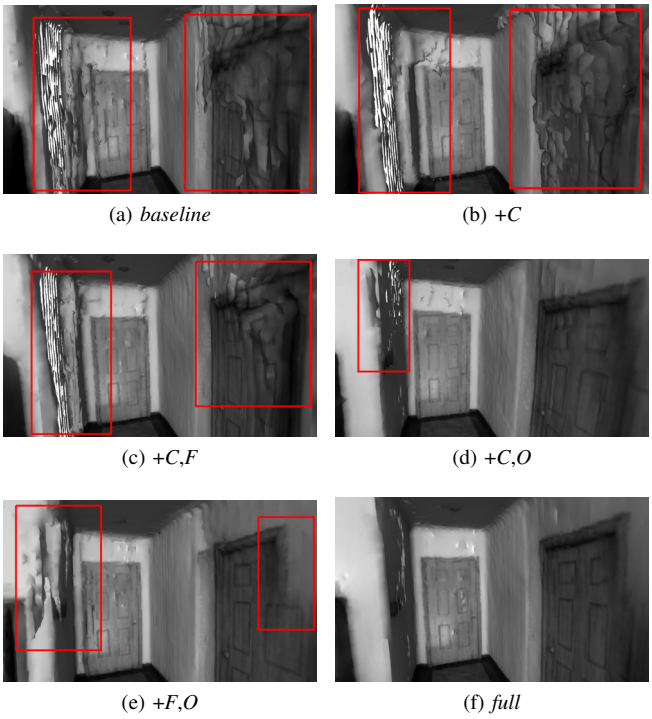
From Table II, it can be seen that virtual normal loss and data augmentation have the greatest impact on depth accuracy and numerical stability improvement. Confidence loss can enhance AbsRel by 5.7%, but the improvement in RMSE is limited, indicating that confidence loss is more helpful for improving accuracy in small error regions. Weight matrix can improve the accuracy but to a lesser extent.

As indicated in Table III, the introduction of each module contributes to the improvement in accuracy. Numerically, the most significant improvement comes from confidence, as expected. Confidence can effectively filter out larger errors over a large area, and this improvement is quite evident in the numerical results. On the other hand, optimization is better suited for fine-tuning depth details, thereby enhancing the overall consistency of the depth. Another conclusion is that noise filtering is crucial. Whether it's the lack of Confidence for noise filtering (+*O, F*) or the absence of an Outlier Removal Strategy (+*C, O*), both can lead to optimization results

TABLE III: Evaluation on depth error on synthetic dataset. The best result is shown in bold. *baseline* stands for directly use depth from network. *C* stands using confidence to remove outliers. *O* stands depth weight optimization. *F* stands for using strategy to filter out outliers. *full* stands for using confidence, outlier removal strategy and depth optimization.

Dataset	*baseline*	*+C*	*+C, O*	*+ C, F*	*+O, F*	*full*
room1_00	0.1341	0.1132	0.1062	0.1065	0.1313	**0.1003**
room1_01	0.1692	0.1489	0.1440	0.1451	0.1634	**0.1280**
room2_00	0.2819	0.2097	0.1840	0.1878	0.2353	**0.1662**
room2_01	0.2017	0.1740	0.1680	0.1702	0.1921	**0.1674**
room3_00	0.3021	0.2838	0.2822	0.2627	0.2911	**0.2550**

struggling to converge to the optimal state. To better visualize the improvement in depth consistency and its impact on the mesh, we have presented various mesh representations in Figure 5. It can be seen that the results on the mesh and numerical accuracy are consistent. Optimization significantly improves depth accuracy and consistency. Effective outlier removal helps optimization converge better, resulting in smoother meshes.

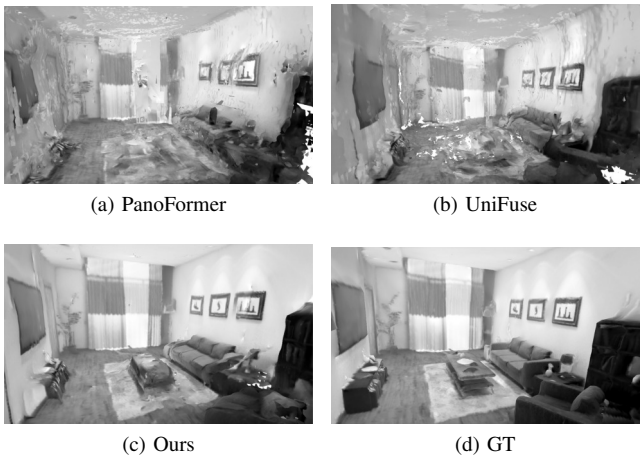


(a) *baseline* (b) *+C* (c) *+C,F* (d) *+C,O* (e) *+F,O* (f) *full*

Fig. 5: The ablation result of meshes on room1_01.

Furthermore, We use Panoformer and UniFuse for depth prediction on the same images and fuse the depths using the same poses. From the results in Figure 6, it's clear that our method significantly outperforms UniFuse and Panoformer.

Real-World Test We validated our method in real-world scenarios and developed a mobile device demo. For real-world tests, the model is trained on the combination of Matterport3d and Pano3d [42]. We not only augment sparse depth by perturbing 10% of depth values, but introduce 2



(a) PanoFormer (b) UniFuse (c) Ours (d) GT

Fig. 6: Meshes generated by different models on room1_01.

pixels of noise to point coordinates. We utilize the official API to transmit online images and IMU data captured by the Insta360 ONEX2 to a smartphone (Mi9) via WiFi. Our SLAM system processed the data transmitted over WiFi in real-time on the Mi9. Due to the time-consuming nature of panoramic stitching algorithms, we can obtain an online image data stream with dual fisheye images only. On the Mi9, the network inference takes averages around 160ms. We run a lightweight optimization online for a small window (taking about 100ms). When the system exits, we execute GBA again to refine all depths. Given the panoramic images' robust scene perception, we augment keyframe selection criteria beyond standard metrics (e.g., feature point tracking count). Specifically, we employ thresholds for keyframe distance and angle, introducing new keyframes only when displacement or rotation exceeds predefined limits. This yields sparser keyframes, ensuring each module meets real-time processing demands efficiently. Similar to [17], the keyframes and the predicted depth optimized by the back-end are sent to a separate TSDF fusion thread to generate the dense mesh. The fusion method is based on the online incremental techniques Mobile3DRecon [43] and Mobile3DScanner [44], which are effective for controlling memory growth and are well-suited for mobile platforms. With a dense mesh, it becomes convenient to implement AR effects on mobile. If applied to robots, it can be used for collision detection or path planning. For more details, please refer to our supplementary video.

V. Conclusion

In this paper, we introduce a novel dense visual-inertial SLAM system based on a back-to-back binocular fisheye data stream. Experiment results demonstrate that our method achieves superior dense mapping results compared to the state of the art. The online demo on mobile verifies the efficiency of our method. In future work, we may explore the use of self-supervised methods to enhance the network's ability to generalize to different environments and scenarios.

REFERENCES

[1] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D lidar inertial odometry and mapping," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3144–3150.

[2] S. Gao, K. Yang, H. Shi, K. Wang, and J. Bai, "Review on panoramic imaging and its applications in scene understanding," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–34, 2022.

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

[4] H.-S. Lin, C.-C. Chang, H.-Y. Chang, Y.-Y. Chuang, T.-L. Lin, and M. Ouhyoung, "A low-cost portable polycamera for stereoscopic 360 imaging," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 4, pp. 915–929, 2018.

[5] R. Komatsu, H. Fujii, Y. Tamura, A. Yamashita, and H. Asama, "360 depth estimation from multiple fisheye images with origami crown representation of icosahedron," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 092–10 099.

[6] S. Xie, D. Wang, and Y.-H. Liu, "OmniVidar: Omnidirectional depth estimation from multi-fisheye images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 529–21 538.

[7] C. Won, J. Ryu, and J. Lim, "OmniMVS: End-to-end learning for omnidirectional stereo matching," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8987–8996.

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

[9] H. Huang and S.-K. Yeung, "360VO: Visual odometry using a single 360 camera," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5594–5600.

[10] H. Seok and J. Lim, "ROVO: Robust omnidirectional visual odometry for wide-baseline wide-fov camera systems," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6344–6350.

[11] R. Liu, G. Zhang, J. Wang, and S. Zhao, "Cross-modal 360 depth completion and reconstruction for large-scale indoor environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25 180–25 190, 2022.

[12] C. Won, H. Seok, Z. Cui, M. Pollefeys, and J. Lim, "OmniSLAM: Omnidirectional localization and dense mapping for wide-baseline multi-camera systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 559–566.

[13] Z. Shen, C. Lin, K. Liao, L. Nie, Z. Zheng, and Y. Zhao, "PanoFormer: Panorama transformer for indoor 360 depth estimation," in *European Conference on Computer Vision*. Springer, 2022, pp. 195–211.

[14] H. Ai, Z. Cao, Y.-P. Cao, Y. Shan, and L. Wang, "HRDFuse: Monocular 360deg depth estimation by collaboratively learning holistic-with-regional depth distributions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 13 273–13 282.

[15] H. Jang, A. Meuleman, D. Kang, D. Kim, C. Richardt, and M. H. Kim, "Egocentric scene reconstruction from an omnidirectional video," *ACM Transactions on Graphics (TOG)*, vol. 41, no. 4, pp. 1–12, 2022.

[16] A. Meuleman, H. Jang, D. S. Jeon, and M. H. Kim, "Real-time sphere sweeping stereo from multiview fisheye images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 423–11 432.

[17] W. Xie, G. Chu, Q. Qian, Y. Yu, H. Li, D. Chen, S. Zhai, N. Wang, H. Bao, and G. Zhang, "Depth completion with multiple balanced bases and confidence for dense monocular slam," *arXiv preprint arXiv:2309.04145*, 2023.

[18] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras, "OmniDepth: Dense depth estimation for indoors spherical panoramas," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 448–465.

[19] C. Zhuang, Z. Lu, Y. Wang, J. Xiao, and Y. Wang, "ACDNet: Adaptively combined dilated convolution for monocular panorama depth estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 3, 2022, pp. 3653–3661.

[20] K. Tateno, N. Navab, and F. Tombari, "Distortion-aware convolutional filters for dense prediction in panoramic images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 707–722.

[21] G. Pintore, M. Agus, E. Almansa, J. Schneider, and E. Gobbetti, "SliceNet: Deep dense depth estimation from a single indoor panorama using a slice-based representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 11 536–11 545.

[22] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai, "BiFuse: Monocular 360 depth estimation via bi-projection fusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 462–471.

[23] H. Jiang, Z. Sheng, S. Zhu, Z. Dong, and R. Huang, "UniFuse: Unidirectional fusion for 360 panorama depth estimation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1519–1526, 2021.

[24] Z. Yan, X. Li, K. Wang, Z. Zhang, J. Li, and J. Yang, "Multi-modal masked pre-training for monocular panoramic depth completion," in *European Conference on Computer Vision*. Springer, 2022, pp. 378–395.

[25] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[26] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[27] S. Sumikura, M. Shibuya, and K. Sakurada, "OpenVSLAM: A versatile visual slam framework," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2292–2295.

[28] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[29] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 15–22.

[30] K. Wu, A. M. Ahmed, G. A. Georgiou, and S. I. Roumeliotis, "A square root inverse filter for efficient vision-aided inertial navigation on mobile devices." in *Robotics: Science and Systems*, vol. 2. Rome, Italy, 2015, p. 2.

[31] H. Seok and J. Lim, "ROVINS: Robust omnidirectional visual inertial navigation system," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6225–6232, 2020.

[32] Z. Wang, K. Yang, H. Shi, P. Li, F. Gao, and K. Wang, "LF-VIO: A visual-inertial-odometry framework for large field-of-view cameras with negative plane," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 4423–4430.

[33] H. Matsuki, R. Scona, J. Czarnowski, and A. J. Davison, "CodeMapping: Real-time dense mapping for sparse slam using compact scene representations," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7105–7112, 2021.

[34] X. Zuo, N. Merrill, W. Li, Y. Liu, M. Pollefeys, and G. Huang, "CodeVIO: Visual-inertial odometry with learned optimizable dense depth," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 14 382–14 388.

[35] M. K. Paul, K. Wu, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "A comparative analysis of tightly-coupled monocular, binocular, and stereo vins," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 165–172.

[36] I.-C. Lo, K.-T. Shih, and H. H. Chen, "Efficient and accurate stitching for 360° dual-fisheye images and videos," *IEEE Transactions on Image Processing*, vol. 31, pp. 251–262, 2021.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[38] W. Yin, Y. Liu, and C. Shen, "Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 7282–7295, 2021.

[39] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and Service Robotics: Results of the 11th International Conference*. Springer, 2018, pp. 621–635.

[40] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang, "Matterport3D: Learning from rgb-d data in indoor environments," *International Conference on 3D Vision (3DV)*, 2017.

[41] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese, "Joint 2D-3D-

Semantic Data for Indoor Scene Understanding," *ArXiv e-prints*, Feb. 2017.

[42] G. Albanis, N. Zioulis, P. Drakoulis, V. Gkitsas, V. Sterzentsenko, F. Alvarez, D. Zarpalas, and P. Daras, "Pano3D: A holistic benchmark and a solid baseline for 360° depth estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 3727–3737.

[43] X. Yang, L. Zhou, H. Jiang, Z. Tang, Y. Wang, H. Bao, and G. Zhang, "Mobile3DRecon: Real-time monocular 3D reconstruction on a mobile phone," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 12, pp. 3446–3456, 2020.

[44] X. Xiang, H. Jiang, G. Zhang, Y. Yu, C. Li, X. Yang, D. Chen, and H. Bao, "Mobile3DScanner: An online 3D scanner for high-quality object reconstruction with a mobile device," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 11, pp. 4245–4255, 2021.